HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Daniel Stephan

Konzeption und Entwicklung von Multitouch-Komponenten unter Verwendung des Frameworks Adobe Flex 4 angewendet in der Simulationsapplikation Cyber

2010

Fakultät **Informations- und Elektrotechnik**

BACHELORARBEIT

Konzeption und Entwicklung von Multitouch-Komponenten unter Verwendung des Frameworks Adobe Flex 4 angewendet in der Simulationsapplikation Cyber

Autor:
Daniel Stephan

Studiengang:
Multimediatechnik

Seminargruppe:
MK07w1-B

Erstprüfer:
Prof. Dr.-Ing. Robert J. Wierzbicki

Zweitprüfer:
Dipl. Med.-Inf. Severin Taranko

Mittweida, 2010

Bibliografische Angaben

Stephan, Daniel: Konzeption und Entwicklung von Multitouch-Komponenten unter Verwendung des Frameworks Adobe Flex 4 angewendet in der Simulationsapplikation Cyber, 85 Seiten, 51 Abbildungen, Hochschule Mittweida (FH), Fakultät Informations- und Elektrotechnik

Bachelorarbeit, 2010

Referat

Diese Bachelorarbeit widmet sich der Erstellung von Komponenten, die auf Multitouch-fähigen Geräten bedienbar sind. Zur Entwicklung dieser Elemente wird das Framework Adobe Flex 4 verwendet. Am Beispiel einer bestehenden Flex-Simulationsanwendung namens Cyber erfolgt die Anwendung dieser Komponenten. Zuvor werden sowohl die Bedienung als auch der Aufbau dieser Anwendung analysiert. Auf der Grundlage dieser Analyse wird ein Konzept erstellt, in dem die Komponenten hinsichtlich ihrer Interaktionsmöglichkeiten und äußerlichen Erscheinung definiert werden. Weiterhin werden die Wege der Umsetzung und Implementierung der Funktionen mit dem Flex-SDK geprüft und erläutert. Dabei sollen vor allem die Handhabung der Touch-Events, aber auch individuelle Gesten sowie die Bedienbarkeit der standardmäßigen Flex-Komponenten näher betrachtet werden.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Abkürzungsverzeichnis

AIR	Adobe Integrated Runtime
API	Application Programming Interface
CAD	Computer Aided Design
CLI	Command Line Interface
DAO	Data Access Object
EOP	End of Production
GP	Große Produktaufwertung
GUI	Graphical User Interface
IOC	Inversion of Control
MIT	Massachusetts Institute of Technology
MSDN	Microsoft Developer Network
NUI	Natural User Interface
ORM	Object-Relational Mapping
OS	Operating System
POI	Point of Information
POS	Point of Sale
RIA	Rich Internet Application
SDK	Software Development Kit
SOP	Start of Production
SQL	Structured Query Language
SWC	Shockwave Flash Component
WAR	Web Application Archive
XLS	Excel Spreadsheet
XML	Extensible Markup Language

1 Einleitung

Berührungsempfindliche Medien nehmen einen steigenden Anteil an unserem alltäglichen Leben ein. Sei es beim Surfen durch Internetseiten unterwegs mit einem neuen Smartphone oder beim Durchblättern der letzten Urlaubsfotos daheim an einem Touchfähigen Tablet - die Steuerung durch Berühren des Bildschirmes wird immer beliebter. Dabei ist der Nutzer zur Bedienung nicht mehr nur noch auf einen Finger begrenzt, sondern kann auf einer Vielzahl von Geräten mehrere Finger zum Navigieren und Interagieren nutzen. Diese Form der Mensch-Maschine-Kommunikation erhöht nicht nur den Unterhaltungswert, sondern soll auch die allgemeine Bedienung gerade für unerfahrene Benutzer erleichtern.

1.1 Problemstellung

Die Unterstützung von Multitouch sowohl von Geräten als auch von der auszuführenden Software ist allerdings nur bedingt verbreitet und noch lange kein Standard. Selbst wenn man ein durch Bildschirmberührungen steuerbares Gerät besitzt, garantiert dies keine intuitivere Bedienung von Anwendungen. Der Benutzer ist zwar in der Lage, Internetseiten und interaktive Applikationen auf seinem Gerät aufzurufen, vorausgesetzt das Betriebssystem und die Hardware ermöglichen es. Doch in den meisten Fällen kann der Benutzer die Anwendungen nur mit einem Finger bedienen. Grund dafür ist zumeist eine nicht vorhandene Unterstützung von Multitouch auf der Softwareseite. Dazu kommt noch der Aspekt, dass Bewegungen wie das Ziehen beziehungsweise Wischen des Fingers von der Anwendung ignoriert werden. Folglich sind die Interaktionsmöglichkeiten auf das Antippen der vorgegebenen Schaltflächen begrenzt. Dieser Umstand zwingt den Benutzer zu dem selben Interaktionsverhalten, das er vom Umgang mit der Maus gewohnt ist. Damit bleiben die Vorteile einer intuitiveren und auf Gesten basierenden Multitouch-Bedienung weiterhin ungenutzt.

Für diesen Sachverhalt bietet Adobe im Bereich der Flash- und Flex-Applikationen seit Juni 2010 mit der Adobe Integrated Runtime 2.0 (AIR)[1] neue Bibliotheken an, um interaktive Anwendungen mit Singletouch- und Multitouch-Unterstützung auszustatten. Viele Projekte haben sich schon vor dieser Veröffentlichung mit Flash und Multitouch beschäftigt und eigene Bibliotheken dazu herausgebracht. Dies erfolgte ohne Verwendung der Flash-internen Berührungsereignisse. Aufgrund des weiten Fortschrittes dieser Projekte wie beispielsweise »Touchlib«[2] und einer großen Anhängerzahl um jene wurden die neuen Flash-Klassen nur wenig zur Kenntnis genommen. Doch die Zahl der Nutzer,

[1] Adobe AIR stellt eine Laufzeitumgebung für Desktopanwendungen dar. Eine ausführliche Erläuterung findet sich im Kapitel *Grundlagen*.

[2] Bei Touchlib handelt es sich um ein Projekt zum softwareseitigem Erkennen und Verarbeiten von Berührungspunkten. Nähere Informationen unter: *http://code.google.com/p/touchlib/*.

die für ihre berührungsempfindlichen Geräte Flash- und Flex-Anwendungen entwickeln wollen, steigt. Denn für den großen Markt der Smartphones ermöglichte Adobe den Entwicklern mit Flash-Professional- und Flex-Erweiterungen AIR-Anwendungen sowohl für Android-Geräte als auch für iPhone und iPad zu realisieren. Zusätzlich soll es mit der Flash Builder-Version »Burrito«[3] in Verbindung mit dem Flex-Framework »Hero«[4] vereinfacht werden für alle bekannteren Plattformen Applikationen zu realisieren: sowohl für mobile Geräte, als auch für Desktop- und Web-Anwendungen. Dafür sind allerdings gewisse Kenntnisse erforderlich, wie Multitouch-Applikationen mit einem verhältnismäßig geringen Aufwand erstellt werden können. Adobe bietet dazu zwar einige Übersichten und eine Einführung an. Aber allein damit und unter Verwendung der standardmäßigen Funktionen wie vordefinierte Gesten ist die Realisierung intuitiv bedienbarer Multitouch-Anwendungen mit Adobe Flex mit großem Aufwand verbunden.

1.2 Zielstellung

Ziel dieser Arbeit ist es, mit Hilfe von Adobe Flex 4 verschiedene Prototypen von Komponenten zu erstellen, die sich hinsichtlich ihrer Erscheinung und Bedienung für den Einsatz auf einem Multitouch-fähigem Gerät eignen. Dies erfordert zum Einen, dass bestimmte Bereiche der Anwendung mit mindestens zwei Fingern bedienbar sind. Zum Anderen muss das berührbare Oberflächendesign an die menschliche Fingergröße angepasst sein.

Die zu entwickelnden Komponenten können dabei auf bestehenden aufbauen oder sich aus gänzlich neuen zusammensetzen. Die Prototypisierung soll auf einem erarbeiteten Konzept basieren, das sowohl die Interaktionsmöglichkeiten als auch das Design der Komponenten genau definiert. Dabei muss stets beachtet werden, dass die Anwendung auch für unerfahrene Benutzer möglichst intuitiv zu bedienen bleibt.

Dieses Vorhaben wird beispielhaft in einer bestehenden Flex-Applikation namens Cyber angewandt. Bei Cyber handelt es sich um eine Simulationsanwendung, die von der *queo GmbH*[5] entwickelt wurde. Die Implementierung von Multitouch-Funktionalitäten in Cyber zieht mehrere Gründe nach sich. So soll die Anwendung zu Präsentationszwecken auf einem großen Touchscreen verwendet werden. Die Interaktion mit grafischen Objekten wirkt durch die Hinzunahme mehrerer Finger und Hände eindrucksvoller und lebendiger. Des Weiteren steigt der Unterhaltungsfaktor beim Bedienen der Anwendung durch diese direkte Interaktion. Zusätzlich dienen die Prototypen als Ausgangspunkte für kommende Implementierungen von Multitouch in Flex-Applikationen. Anwendungsgebiete im mobilen Bereich aber auch auf Multitouch-fähigen Desktop PCs sind dabei denkbar.

[3] Mit Burrito wird die nachfolgende Version des aktuellen Flash Builder 4 bezeichnet.
[4] Hero kennzeichnet die kommende Version des Flex Software Development Kits.
[5] Bei der queo GmbH handelt es sich um eine Crossmedia-Agentur aus Dresden. Nähere Informationen unter: *www.queo-media.com*.

Im Vergleich zu Multitouch-unterstützenden Flash- und Flex-Applikationen, die beispielsweise im Rahmen von Projekten wie »Touchlib« entstanden, werden sich die Komponenten dieser Bachelorarbeit auf die Flash-internen Touch-Events beziehen. Dabei soll untersucht werden, wie diese Events nutzbringend unter Berücksichtigung der Anforderungen an die Komponenten eingebracht werden können. Zusätzlich werden bestehende Flex-Standard-Komponenten auf eine Bedienung mittels Fingerberührungen geprüft. Auf der Grundlage dieser Ergebnisse sollen Entwickler von Multitouch-fähigen Flex-Anwendungen ein besseres Verständnis für deren Handhabung in Verbindung mit berührungsempfindlichen Geräten bekommen.

1.3 Aufbau der Arbeit

Im folgenden, zweiten Kapitel *Grundlagen* werden zunächst wichtige Voraussetzungen für das Verständnis der Multitouch-Technologie erläutert. Dabei stehen sowohl aktuelle Geräte mit Multitouch-Unterstützung als auch deren Anteilnahme im Alltag im Fokus. Weiterhin findet der Leser eine Erläuterung der Multitouch-Funktionalität am Beispiel eines All-In-One-PCs namens xDesk. Dieses Gerät stellte eines meiner für die Bachelorarbeit verwendeten Testgeräte dar und kann eine Vielzahl an gleichzeitigen Berührungen verarbeiten. Mit der Vorstellung von relevanten Gesten und Problemen, die berührungsempfindliche Geräte mit sich bringen, werden darüber hinaus grundlegende Themen beschrieben, die für die weiteren Abschnitte der Bachelorarbeit von Bedeutung sind. Abschließend werden in diesem Kapitel das Flex-Framework sowie dessen Zusammenhang zu Adobe AIR und die für die Erstellung der Komponenten gewählte Entwicklungsumgebung näher erläutert.

Damit in einem späteren Schritt ein Konzept für Komponenten in Cyber erstellt werden kann, bedarf es zunächst einer genauen Analyse dieser Anwendung. Diese findet sich im dritten Kapitel *Analyse* wieder. Darin erfährt der Leser wichtige Sachverhalte bezüglich des Aufbaus und der Funktionsweise des Front-Ends und Back-Ends von Cyber. Außerdem wird die Bachelorarbeit auf einen Bereich des Front-Ends begrenzt, der auf all seine Funktionen und Komponenten untersucht wird. Als Grundlage für spätere Vergleiche und Ausgangpunkt für das Konzept wird die bisherige Bedienung dieses Bereiches genau analysiert.

Das Konzept für den Komponentenaufbau wird im vierten Kapitel *Konzept* aufbauend auf den Erkenntnissen der Analyse entwickelt. Die in *Grundlagen* betrachteten Themen zu Gesten und auftretenden Problemen üben dabei zusätzlichen Einfluss aus. In dem Konzept wird für eine bessere Übersicht der betrachtete Teil von Cyber nochmals in drei miteinander verbundene Bereiche geteilt. Jeder dieser Bereiche stellt Komponenten dar, deren Interaktionsmöglichkeiten und Design definiert werden. Zusätzlich findet der Leser Ansätze für die Umsetzung dieser Vorgaben mit Hilfe von Flex-internen Programmiersprachen.

Anschließend wird im fünften Kapitel *Implementierung* auf die Einfügung der entwickelten Komponenten in die Cyber-Anwendung eingegangen. Dabei erhalten sowohl die Einspeisung erstellter Klassen als auch externe Bibliotheken einen Fokus.

In dem abschließenden Kapitel *Zusammenfassung* findet der Leser einen Vergleich zwischen neuer und alter Bedienung von Cyber. Außerdem werden wesentliche Erkenntnisse hervorgehoben, die diese Bachelorarbeit betreffen. Gerade auf Probleme und deren Lösungen, aber auch auf eventuelle Weiterentwicklungen wird näher eingegangen.

2 Grundlagen

2.1 Multitouch

Multitouch definiert eine Technologie, bei der die sensitive Oberfläche eines Eingabegerätes, das auch gleichzeitig Ausgabemedium sein kann, mit mehr als einem Finger bedient werden kann. Damit werden die taktilen und visuellen Wahrnehmungssysteme des Menschen angesprochen. Geräte, die eine Touch-Oberfläche aufweisen, wurden bislang meist zu den Single-Touch-Geräten gezählt. Denn ähnlich wie bei der Steuerung mit der Maus, waren diese Geräte in der Lage lediglich einen Cursor beziehungsweise einen Berührungspunkt zu unterstützen. Mit der Unterstützung mehrerer Berührungen ändert sich natürlich auch das Verhalten des Nutzers. So kann der gewohnte Single-Drag-And-Drop[6] zum gleichzeitigen Verschieben verschiedener Komponenten mit mehreren Fingern erweitert werden. Bilder sind mit zwei Fingern skalier- und rotierbar. Mehrere Benutzer sind in der Lage eine Anwendung zu bedienen.

Doch anhand dieser Möglichkeiten muss auf Softwareseite ein darauf abgestimmtes Bedienkonzept vorliegen. Dabei rücken mittlerweile die sogenannten Natural User Interface-(NUI) Bedienkonzepte zunehmend in den Mittelpunkt. Hauptmerkmal solcher Interfaces ist die Bedienung einer Maschine durch mindestens einen menschlichen Sinn. Diese sollen dabei helfen, weg von der herkömmlichen Bedienung eines Computers mit Maus und Tastatur hin zur Interaktion mit den eigenen Händen zu gelangen [15]. Bisher galt nach der Ablösung des Command Line Interfaces (CLI), welches die Eingaben mit Hilfe der Tastatur und mittels textbasierter Codes ermöglichte, das Graphical User Interface (GUI) als Standard bei der Bedienung von Computern [33]. Mit dessen Erweiterung um grafische Inhalte wurde erstmals eine Steuerung im zweidimensionalen Raum unter Einsatz der Maus ermöglicht. Lange Zeit wurde dies nun als herkömmlicher und gewohnter Weg der Steuerung bei dem alltäglichen Gebrauch von Computern angesehen. Touchscreens und Tablet-PCs brachten zwar frische Denkanstöße für neue Bedienkonzepte, ermöglichten aber ähnlich wie beim Umgang mit dem PC nur die Steuerung mit einem Cursor beziehungsweise einem Berührungspunkt. Dadurch blieb die Navigation durch Programme oder Internetseiten nahezu die gleiche.

Nun sollen neue Bedienkonzepte die Steuerung für den Endbenutzer effizienter, intuitiver und simpler gestalten. Doch dazu müssen neben der Hardware auch auf Softwareseite einige Bereiche überdacht und neu entwickelt werden. Dabei ist es nicht ausreichend, aktuelle Internetseiten oder Flash-Inhalte mit einem Multitouch-fähigen Gerät zu bedienen, sondern diese auch auf die entsprechenden Geräte anzupassen. Denn das Potential, das sich hinter der gesamten Technologie versteckt, ist groß und Multitouch

[6] Single-Drag-And-Drop bezeichnet die Verschiebung und Platzierung eines Objektes mit einem Cursor.

bietet viele Möglichkeiten dieses auch auszuschöpfen.

2.1.1 Gesten

Multitouch erlaubt dem Benutzer zahlreiche verschiedene Möglichkeiten zur Bedienung eines Gerätes. Allerdings sind die Möglichkeiten abhängig von der Anzahl der maximal verarbeitbaren Berührungspunkte des Gerätes. Dadurch ist es ein Unterschied, ob der Nutzer mit einem Microsoft Surface[7] arbeitet, der über 50 Berührungspunkte unterstützt oder ob der Benutzer ein Gerät mit nur zwei erkennbaren Berührungspunkten benutzt. Ein oft angesprochener Bereich innerhalb von Multitouch sind die Gesten. In diesem Kontext betrachtet, fassen Gesten mehrere in einer bestimmten Reihenfolge auftretenden Multitouch-Ereignisse oder -berührungen zusammen und dienen dadurch zum Aufrufen einer bestimmten Funktion.

Eine weit verbreitete Geste ist die Zoom- oder auch Pinch-Geste, die der Nutzer mit zwei Fingern, wie in Abbildung 2.1 nachstellen kann. Dabei werden die Finger um ein Bild zu vergrößern voneinander weggestreckt oder um es kleiner erscheinen zu lassen, zueinander bewegt. Mit der Rotieren-Geste können Objekte um einen der beiden Berührungspunkte gedreht werden. Um dies zu erreichen, müssen zwei Finger aufliegen und einer von beiden so bewegt werden, dass er sich kreisförmig um den anderen bewegt. Beide Gesten eignen sich hervorragend für die Arbeit mit grafischen Objekten und sind vielen Anwendern bekannt. Durch dieses Wissen lassen sich Programme noch intuitiver bedienen, da der Nutzer nicht erst die Schaltfläche oder die Taste zum Drehen eines Objektes heraussuchen muss.

Abbildung 2.1: Links - die Zoom-Geste, rechts - Two-Finger-Rotate; Quelle: [18]

Für den Fall, dass man beispielsweise in einem Textfeld scrollen muss, um weiteren Text erscheinen zu lassen, gibt es die Scroll-Geste. Wie in Abbildung 2.2 angedeutet, kann diese ebenfalls mit zwei Fingern erzeugt werden. Dabei setzt der Anwender beide Finger in das Textfeld und bewegt sie möglichst parallel vertikal im Feld nach oben

[7] Geräte der Serie Microsoft Surface entsprechen 30 Zoll breiten Multitouch-fähigen All-In-One-PCs.

oder unten. Analog dazu kann die Geste auch horizontal eingesetzt werden. Diese Bewegung kommt einem Ziehen oder Verschieben des Textes gleich. So kann auf den Scrollbalken gänzlich verzichtet werden. Man ist dadurch nicht mehr darauf angewiesen, den Cursor zum Scrollbalken bewegen und mit dessen Verschiebung den Text zu navigieren.

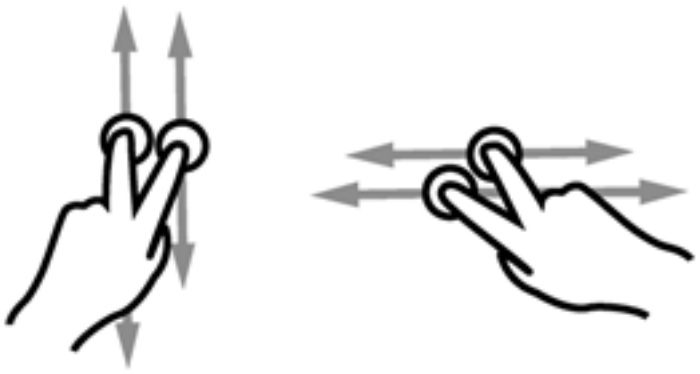

Abbildung 2.2: Die Zwei-Finger-Scroll-Geste: links vertikal - rechts horizontal; Quelle: [18]

Ebenso gibt es Alternativen für Navigationsschaltflächen, mit denen der Nutzer beispielsweise in einer Fotogalerie zum nächsten oder zum vorherigen Bild springen kann. Dieselbe Funktion kann mit der sogenannten Flick- oder Swipe-Geste erreicht werden: mit mehreren aufgesetzten und schnell nach links oder rechts bewegten Finger könnte hier der Benutzer die Anwendung steuern. Ähnliche Interaktionsmöglichkeiten kennen Benutzer eines Gesten-unterstützenden Mac Books bereits. Dort kann das beliebte Wischen mit den Fingern zum Weiterspulen in Videos, Blättern durch Dokumente oder Wechsels des aktuellen Programmes eingesetzt werden [25]. Dadurch ist eine Reduzierung künftiger Navigationsschaltflächen auch für Web-Applikationen vorstellbar. Denn auch bei dieser Geste ist der Nutzer nicht erst darauf angewiesen, seinen Cursor an eine Schaltfläche zu bewegen, die ihm die nächste Seite anzeigt. Dieses intuitive Nutzerverhalten reduziert den Aufwand eine spezielle Funktion anzuwählen und ist äußerst effektiv bei der Bedienung von Programmen.

Ein weiterer Betrachtungspunkt betrifft das Auswählen von Objekten, das einem Klick mit der linken Maustaste gleichkommt. Wie wird dies realisiert, wenn bei der Bedienung einer berührungsempfindlichen Oberfläche ständig die Finger darauf entlang fahren und damit immer einen Klick auslösen müssten? Eingesetzt wird dafür oftmals die Tap-Geste. Dabei berührt man kurz mit einem Finger die Oberfläche an der Stelle des auszuwählenden Objektes. Man tippt es also an. Mit dem sogenannten Douple-Tap sind auch Doppelklicke nachstellbar. Auch bei den Tap-Gesten ist es nicht unüblich, diese für zwei Finger auszulegen.

Wenn an Touch-Funktionen gedacht wird, spielt auch das Verschieben von Objekten entlang der Oberfläche eine wichtige Rolle. Beispielsweise wenn mehrere Bilder übereinander liegen und diese, um die darunter liegenden Objekte zu sehen, in verschie-

dene Richtungen geschoben werden. Ähnlich wie eine gedrückte linke Maustaste kann der Benutzer dies auch mit der Drag-Geste erreichen. Wie in Abbildung 2.3 erkennbar, drückt er einen Finger auf das zu bewegende Objekt und bewegt ihn ohne die Berührung zu unterbrechen. Diese Drag-Geste kann auch für mehrere Finger definiert werden. Dabei sollte immer bedacht werden, ob sich der Einsatz mehrerer Finger oder Hände für die zu entwickelnde Anwendung lohnt. Vor allem für grafische Visualisierungen sind Fünf-Finger-Rotationen, -Verschiebungen oder -Vergrößerungen gern gesehene Interaktionsmittel.

Abbildung 2.3: Verschiebungen: links - One-Finger-Drag, rechts - Two-Finger-Swipe; Quelle: [18]

Dies war nur ein Überblick über aktuell weit verbreitete Gesten. Natürlich sind noch viele weitere im Einsatz, die mittlerweile auch am heimischen PC entwickelt und auf einem Multitouch-fähigen Gerät verwendet werden können. Vor allem Apple legt mit seinen zahlreichen Gesten-Patentierungen hohen Wert auf stetige Erweiterungen in diesem Bereich. Diese gehen mittlerweile so weit, dass alltägliche Funktionen wie Kopieren, Ausschneiden oder dem Aufrufen des Tabulators Verwendung finden.

2.1.2 Funktionsweise am Beispiel xDesk

Der xDesk des deutschen Unternehmens impressx ist ein Vertreter sogenannter Multitouch-fähiger All-in-One-PCs und diente während dieser Arbeit als Testumgebung für die entwickelten Komponenten. Mit seinem 52 Zoll Multitouchscreendisplay können technisch gesehen mehrere Personen gleichzeitig das Gerät bedienen, da er in der Lage ist, maximal 127 Berührungspunkte zu verarbeiten. Wie in Abbildung 2.4 liegt das Display auf einem rechteckförmigen Körper geschützt durch eine Glasoberfläche. Im Inneren des xDesk befindet sich ein lauffähiger PC mit dem Betriebssystem Windows 7. Damit ist der xDesk voll funktionsfähig und benötigt zum Betrieb keine externen Geräte. Optionale Komponenten wie Maus und Tastatur sind aber weiterhin anschließbar.

Mittels einer im Inneren des Gerätes angebrachten Infrarotoptik registrieren Sensoren alle auf der Touch-Oberfläche abgegebenen Berührungen und Gesten. Dieses Verfahren findet unter Anwendung der Diffused Illumination, also einer indirekten Beleuchtung, statt. Haupteigenschaft dieser Technik ist die Beleuchtung infraroten Lichtes an die Unterseite der Touch-Oberfläche [7]. Weiterhin bricht, wie in Abbildung 2.5 zu sehen, ein

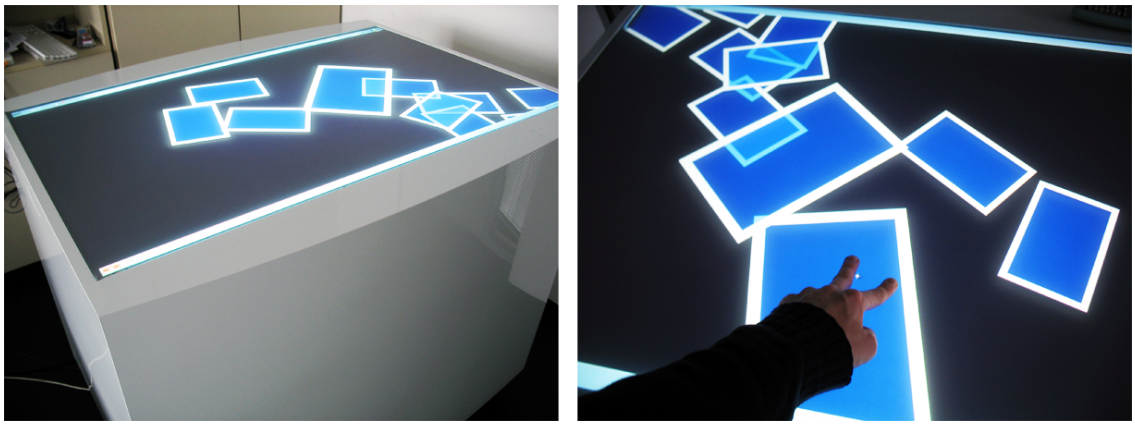

Abbildung 2.4: Der Multitouch-fähige xDesk

Diffuser das auf die Oberfläche treffende Licht. Sobald ein Objekt die Oberfläche berührt, wird dadurch mehr Licht reflektiert als von dem Diffuser oder von Hintergrundobjekten. Dieses zusätzliche Licht wird am Beispiel des xDesk von zwei Kameras aufgenommen und anschließend interpretiert [19]. Die Auswertung erfolgt im Rechner.

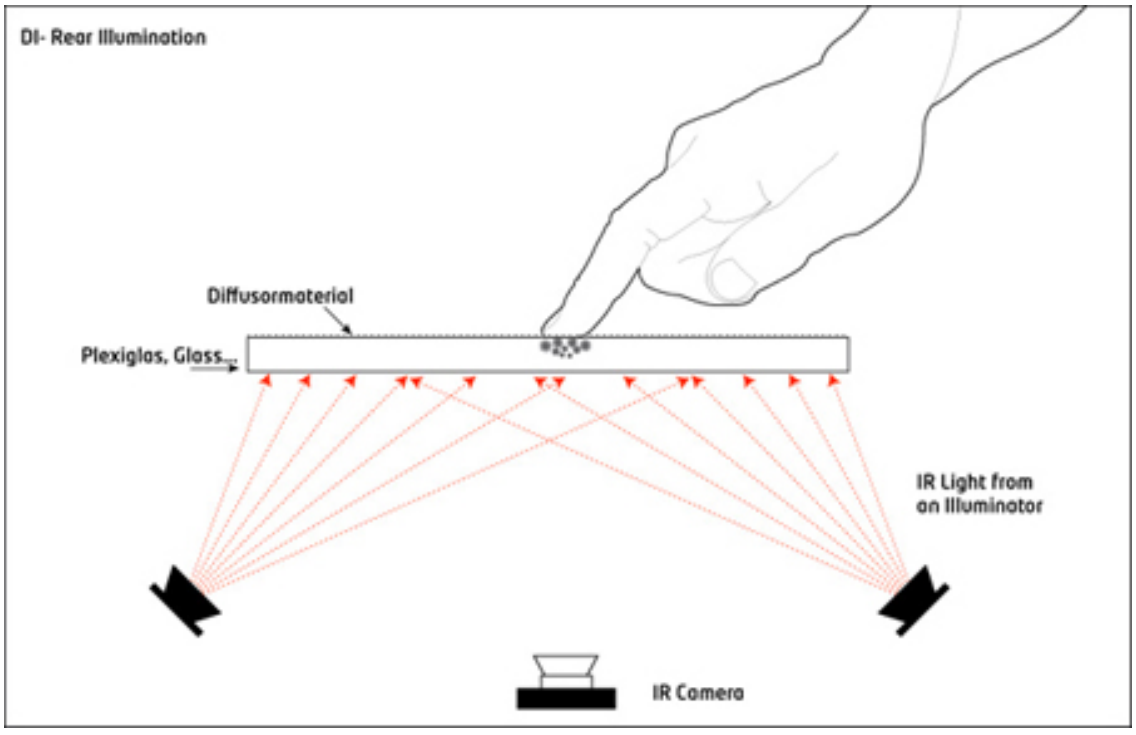

Abbildung 2.5: Funktionsweise der Diffused Illumination bei der Berührung mit einem Finger; Quelle: [7]

Damit bei dem Messvorgang keine Objekte erfasst werden, welche die Oberfläche nicht berühren, arbeitet man beim Tracking[8] mit speziellen Filtern. Diese ermöglichen auch eine Verwendung bei verschiedenen Lichtverhältnissen. Damit der Benutzer aber erst ein Bild an der Oberfläche sieht, bedarf es diverser Projektionstechnik. Dabei kommt ein

[8] Mit Tracking wird das Verfolgen und Erfassen der Lichtsignale beschrieben.

3LCD-Projektor[9] von Epson zum Einsatz. Dieser ist in der Lage die projizierten Bildinformationen, welche er über den eingesetzten Rechner erhält, in einer hohen Bildqualität wiederzugeben. Mittels der zusätzlich implementieren Produkterkennung können verschiedene, mit einem Barcode gekennzeichnete Objekte auf die Oberfläche gelegt und erkannt werden. Diese Funktion erleichtert sowohl die Darstellung von Produktinformationen als auch Vergleiche verschiedener Artikel.

2.1.3 Aktuelle Plattformen

Mit der zunehmenden Marktanteilnahme von Geräten mit berührungsempfindlichen Bildschirmen nehmen diese einen immer größer werdenden Stellenwert in unserem Alltag ein. Sei es privat bei der Verwendung neuer Smartphones oder in Unternehmen bei Produktpräsentationen anhand großer Touchscreen-Wände. Singletouch- und Multitouch-Geräte begegnen einem nicht mehr nur noch auf Fachmessen und Ausstellungen sondern immer öfter im alltäglichen Leben. Neben Apple und derem erfolgreichen Einbringen mobiler Multitouch-fähiger Geräte in den Mainstream gewinnen Unternehmen wie Microsoft oder HP mit ihren Produkten zunehmend an Marktanteil. Eine Nielsen-Studie über Smartphone-Verkäufe in den USA von Oktober 2010 bestätigt, dass Geräte mit einem iPhone Operating System (OS) nur noch 25% des dortigen Marktes ausmachen. Smartphones mit einem Android-Betriebssystem haben hingegen in dem letzten Jahr um 18 Prozentpunkte auf 32% des Marktes aufgeschlossen [8].

Einer der bekanntesten Vertreter von mobilen Multitouch-Geräten ist das von Apple entworfene und vermarktete iPhone. Dieses ist derzeit in der Version 4G erhältlich und erkennt bis zu zwölf gleichzeitig wirkende Berührungen [33]. Verschiedene Gesten zum Zoomen und Schieben von grafischen Objekten gehören zu den beliebten Interaktionsmöglichkeiten. Dazu sollen bald zusätzliche Gesten wie einfache *Copy and Paste*-Funktionen stoßen, die anhand bestimmter Fingerbewegungen realisiert werden. Neben dem iPhone steht auch das iPad seit Längerem im Fokus der Öffentlichkeit und sticht durch seine intuitive Bedienung hervor. Es kann bis zu elf Berührungen gleichzeitig wahrnehmen und erkennt dieselben Gesten wie das iPhone. Ihre kapazitiven[10] Bildschirmoberflächen können zwar mit Fingern, aber nur mit wenigen Eingabestiften bedient werden. Im Bereich der Notebooks könnte Apples MacBook Air ebenfalls zu den Multitouch-Vertretern gezählt werden. Denn mit dessen Multiouch-Trackpad sind Vier-Finger-Gesten möglich, um beispielsweise Bilder oder die Schriftgröße im Browser zu vergrößern oder verkleinern. Allerdings sind die Gesten auf einige wenige begrenzt und funktionieren nur in bestimmten Anwendungen. Unter einigen Experten wird das MacBook Air hingegen nicht als echtes Multitouch-Gerät angesehen, da Eingabegerät und Ausgabedisplay getrennt auftreten.

[9] 3LCD kennzeichnet eine Projektionstechnologie bei der die auf 3-Chip basierende bildgebende Einheit sehr natürliche Bilder erzeugen kann.

[10] Kapazitive Touchscreens ermitteln die Position der Berührung anhand resultierender Ströme in einem elektrischen Feld.

Im Bereich der Desktop PCs gewinnt HPs *Touch Smart*-Serie zunehmend an Bedeutung. Mit der Einführung von Windows 7 sind diese auch erstmals von der Seite des Betriebssystems aus Multitouch-fähig. Vorher wurde lediglich die Single-Touch-Funktion standardmäßig unterstützt. Mit der Einführung des neuen Betriebssystems sind auch auf Geräten, wie dem Touch Smart 600, unter Anderem Gesten zum Bearbeiten von Fotos mittels mehrerer Finger möglich. Zur Bedienung können jederzeit Tastatur, Maus oder spezielle Eingabestifte zusätzlich herangezogen werden. Gänzlich ohne Tastatur und Maus kommt hingegen das von Microsoft entwickelte Produkt Microsoft Surface aus. Microsoft beschreibt auf deren Webpräsenz die neuen Möglichkeiten, auf die ein Surface-Benutzer trifft, mit: „Surface computing breaks down traditional barriers between people and technology, changing the way people interact with all kinds of everyday information - from photos to maps to menus“ [10]. Es grenzt sich also deutlich von den traditionellen Interaktionstechniken ab und wird von Microsoft auch als Gerät zum Umgang mit alltäglichen elektronischen Informationsquellen gesehen. Wie in Abbildung 2.6 zu sehen, besteht das Tisch-ähnliche Produkt aus einer 30 Zoll Tischplatte und einer darunterliegenden Box. Im Inneren befinden sich fünf Kameras zur Aufzeichnung der durch Berührung ausgelösten Infrarot-Reflexionen. Mit 52 gleichzeitig erkennbaren Berührungspunkten gehört es zu den wenigen Vertretern, die mehr als 5 Personen gleichzeitig das Gerät bedienen lassen können [33].

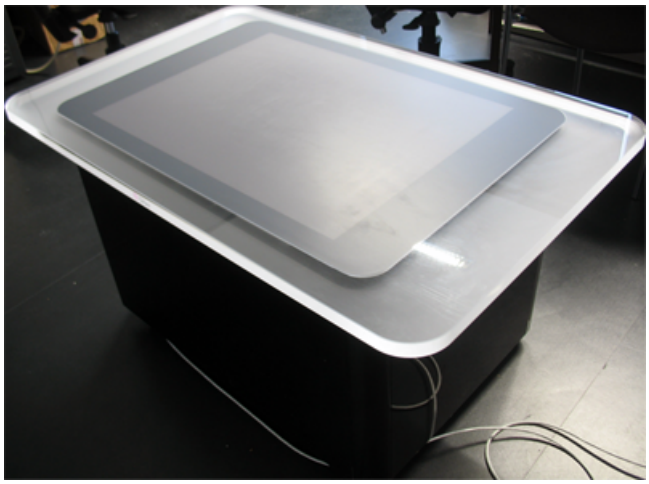

Abbildung 2.6: Der Microsoft Surface kann bis zu 52 Berührungen gleichzeitig erkennen

Neben den All-In-One-PCs und Laptops gibt es auch Multitouch-fähige Displays. Ein Vertreter dieser Gruppe ist das M2256PW von 3M. Es unterstützt 20 gleichzeitig auftretende Fingerberührungen und besitzt mit einer Ansprechzeit von 6 ms sehr gute Reaktionszeiten [17]. Durch seine Kompatibilität mit Windows 7 kann es neben dem Heimeinsatz auch für professionelle Arbeiten wie CAD-Anwendungen[11] eingesetzt werden und dort die Arbeit effizienter gestalten. Im Rahmen dieser Bachelorarbeit stand das M2256PW neben dem xDesk als Testgerät zur Verfügung.

[11] CAD oder auch Computer Aided Design beschreibt die Konstruktion meist technischer Bilder mit Hilfe eines Computers.

Im aufstrebenden Bereich der Smartphones gibt es eine Reihe an Produkten, welche ebenfalls mit Multitouch-Funktionalitäten ausgestattet sind. Abbildung 2.7 zeigt zwei dieser Vertreter. So können Geräte wie Googles Nexus One oder das Desire von HTC durch Fingerberührungen auf deren Displayoberfläche bedient werden. Als Betriebssystem wird dabei Android eingesetzt, welches seit der Version 2.0 Multitouch unterstützt [5]. Außerdem gibt es weitere Hersteller von Multitouch-fähigen Smartphones, Tablets oder Notebooks, die den Markt mit neuen Alternativen füllen.

Abbildung 2.7: Multitouch-fähige Smartphones: links - Googles Nexus One; Quelle [20], rechts - das Desire von HTC; Quelle [9]

2.1.4 Anwendungsfälle im Alltag

Touchscreens begegnen uns immer häufiger im Alltag. Dabei stellen sie die Schnittstelle zwischen dem Benutzer und der ausgeführten Software samt dem umgebenden System dar. Mit ihrer Hilfe gelingt es, die Verbindung zwischen Computer und Mensch zunehmend enger werden zu lassen. Dies wird dadurch gewährleistet, dass die Anwendungen direkt auf der Bildschirmoberfläche mit den Fingern oder speziellen Stiften gesteuert werden können. Dabei sieht der Benutzer an den Stellen, die berührt wurden eine sofortige Antwort des Systems. Sei es durch Hervohebungen der gedrückten Schaltflächen oder durch verschiebbare Objekte, deren Position sich an der aktuellen Fingerlage richtet. Da sich die Bedienung solcher Software nicht nur als intuitiver erweisen kann, sondern meistens auch einen hohen Unterhaltsungswert mit sich bringt, gibt es sowohl für Single- als auch Multitouch viele Anwendungsgebiete.

Oftmals findet man berührungsempfindliche Bildschirme an öffentlichen Orten, wie Museen oder Bahnhöfen, in Form von Terminals oder auch Kiosksystemen vor. Mit der Möglichkeit diese mit einem Finger zu bedienen, können sich Besucher oder Passanten nähere Informationen zu einem für sie interessanten Thema anzeigen lassen. Solche Vertreter der sogenannten »Points of Information« (POI) oder auch Punkte zur Informationsvermittlung werden in diesem Fall durch spezielle Multimedia-Systeme reali-

siert [27]. Auf der Grundlage einer möglichst intuitiven Bedienung kann sich der Benutzer schnell zu den gewünschten Informationen bewegen, die meist auf die entsprechende Zielgruppe eingestellt ist. Mit Mehrfinger-Bewegungen könnte beispielsweise die Navigation durch mehrere Seiten vereinfacht werden, sodass statt dem Berühren einer Navigationsschaltfläche mit einer Wisch-Geste die nächste Seite angezeigt wird.

Neben dem Punkt zur Informationsbeschaffung ist auch der Ort des Einkaufs, der »Point of Sale« (POS), durch diese Geräte beeinflussbar. Dies ist der Ort, an dem der Kunde unmittelbaren Kontakt zu der erwerbbaren Ware hat [23]. Um den Kauf der Ware voran zu treiben, werden natürlich einige Maßnahmen von der Unternehmensseite aus ergriffen. Somit gestalten elektronische Shopsysteme den POS im heimischen Bereich immer attraktiver. Aber auch unterwegs kann dem Einkaufswunsch nachgegangen werden - vorausgesetzt man ist im Besitz eines entsprechenden Smartphones oder ähnlichen Gerätes. Mit der Möglichkeit, solche Shops nahezu überall zu bedienen und dies auch unabhängig von der Hardware, vergrößert sich die potenzielle Kundschaft maßgeblich. Doch nicht nur am privaten Computer oder auf Smartphones kann der Kunde auf die Touch-Technologie zurückgreifen, sondern auch in dem betreffenden Laden selbst. Vor allem beim Kauf physikalischer Objekte wie Möbel oder Elektrogeräte zahlt sich der Einsatz von Touchscreens aus. So sind Anwendungen denkbar, in denen der Kunde das gewünschte Objekt im dreidimensionalen Raum nach Belieben drehen und rotieren kann. Diese Funktionen würden auf der Grundlage von Multitouch gerade mit mehreren Fingern einfach auszuführen sein, was den Kunden noch vor dem Kauf mit der Ware vertraut macht. Ebenso wären Käufer und Verkäufer gleichzeitig in der Lage dasselbe Gerät zu bedienen. Dies bringt den großen Vorteil mit sich, dass der Kunde seine Wünsche nicht erst übermitteln muss, sondern diese direkt vor dem Verkäufer umsetzen kann. Die gemeinsame Handhabung solcher Geräte würde die gewohnte Kaufatmosphäre auflockern, die Kommunikation zwischen beiden Seiten verbessern und dabei helfen die Wünsche des Kunden zu erfüllen.

Ähnliche Ziele verfolgt der »Product Navigator« von BMW in München. Diese Software wird auf einem Microsoft Surface eingesetzt und dient dem BMW-Interessenten, sein gewünschtes Auto nach Belieben anzupassen [4]. Dabei kann neben der Lackierung auch die Innenausstattung mit Fingerberührungen eingestellt werden. Zur besseren Visualisierung der verwendbaren Farben sind die Lackiermöglichkeiten mit flachen, plättchenförmigen Tafeln auswählbar. Diese können auf die Oberfläche des Microsoft Surface gelegt werden, wodurch sich das Auto umgehend neu einfärbt. Die Erkennung der Plättchen erfolgt durch einen an der Unterseite angebrachten Barcode [33]. Solche Szenarien lassen sich ebenfalls auf Möbelhäuser und sonstige Warenhäuser übertragen.

Ein hohes Potenzial für Touch-Anwendungen bringt auch die Spieleindustrie mit sich. Bislang stehen dabei Applikationen für Smartphones im Vordergrund. So gibt es in den Genrebereichen der Geschicklichkeits- und Jump & Run-Spiele zahlreiche Applikationen, die größtenteils mit Hilfe von einem Finger gesteuert werden. Diese erleichtern

zumeist durch eine einfache Bedienung sowie geringe Menüfelder einen schnellen Einstieg in das Spiel. Doch auch für PC-Spieler mit Touchscreens vergrößert sich das Angebot an möglichen Spielen. Der französische Spieleentwickler Ubisoft hat mit »RUSE« 2010 eines der ersten Multitouch-fähigen Strategiespiele für den PC veröffentlicht. Wie in Abbildung 2.8 zu sehen ist, kann sich der Spieler mit Zoom- und Drag-Gesten frei durch die virtuelle Welt bewegen. Die Schaltflächen des Menüs sind auf ein Minimum begrenzt und in ihrer Größe an Touchscreens angepasst. Durch diese Form der Bedienung sind völlig neue Spielprinzipien denkbar. Mehrere Spieler können zusammen an einem Tisch oder Bildschirm ihre Eingaben betätigen, was den Unterhaltungswert spürbar steigen lassen würde. Mit diesen neuen Interaktionsmöglichkeiten kann ein Spieler bestimmte Anweisungen auf eine intuitivere Art geben. Dies erhöht die Möglichkeiten, sich in einem Spiel zu bewegen oder bestimmte Objekte zu steuern.

Abbildung 2.8: Das Multitouch-fähige Strategie-Spiel »RUSE«; Quelle [13]

Neben der Unterhaltungsindustrie profitiert auch die Kommunikation innerhalb eines Unternehmens von der Touch-Technologie. Bei Meetings oder Gruppengesprächen lässt sich beispielsweise die Präsentations- und Arbeitsoberfläche von mehreren Personen gleichzeitig bedienen. Durch die gemeinschaftliche Bedienung ist ein besserer Austausch zwischen den Kollegen denkbar. Ebenso kann sich jeder Beteiligte am Tisch die dargestellten Objekte so drehen, dass er sie bestmöglich sehen kann. In diesem Bereich der unternehmensinternen Simulationsanwendungen werden sich auch die Komponenten wiederfinden, die in dieser Bachelorarbeit entwickelt werden.

Die hier aufgelegten Darstellungen sind allesamt Beispiele für einen sinnvollen Einsatz der Singletouch und Multitouch-Technologie im alltäglichen Leben. Es bleibt abzuwarten, inwiefern sich Multitouch in den genannten Bereichen einsetzen und vor allem durchsetzen kann. Doch auf Grundlage geeigneter Bedienkonzepte kann es sowohl bei der Interaktion mit Geräten als auch bei der menschlichen Kommunikation zu vielen positiven Weiterentwicklungen führen.

2.1.5 Auftretende Probleme

Neben all den positiven Erscheinungen bringt die Multitouch-Technologie derzeit noch einige Probleme mit sich. Auf Touch-fähigen Geräten kann es durchaus vorkommen, dass bei der Bedienung von Software Schwierigkeiten mit dem Interfacedesign auftreten. Ein Ärgernis findet sich in nicht angepassten Bedienoberflächen wieder. Kleine Schaltflächen und ein nicht für derartige Bedienungen ausgelegtes Oberflächendesign erschweren die Benutzerfreundlichkeit. Denn der Finger eines Menschen übertrifft die Ausmaße des pfeilförmigen Cursors bei Weitem. Abhilfe bringt dabei zumeist eine Vergrößerung des aktuellen Bildausschnittes oder die Verwendung eines Eingabestiftes, wobei beide Möglichkeiten einer optimalen Anwendung der Multitouch-Technologie eher im Wege stehen. Microsoft hat beispielsweise das Multiouch unterstützende Betriebssystem Windows 7 für solche Anwendungsfälle optimiert. Bedienelemente und Layout der Benutzeroberfläche ändern sich, wenn das System Berührungen empfangen soll. Damit vergrößern sich Abstände zwischen Schaltflächen, die berührbare Fläche von Buttons wird erhöht und Menüelemente ordnen sich wie in Abbildung 2.9 in großzügiger Form an.

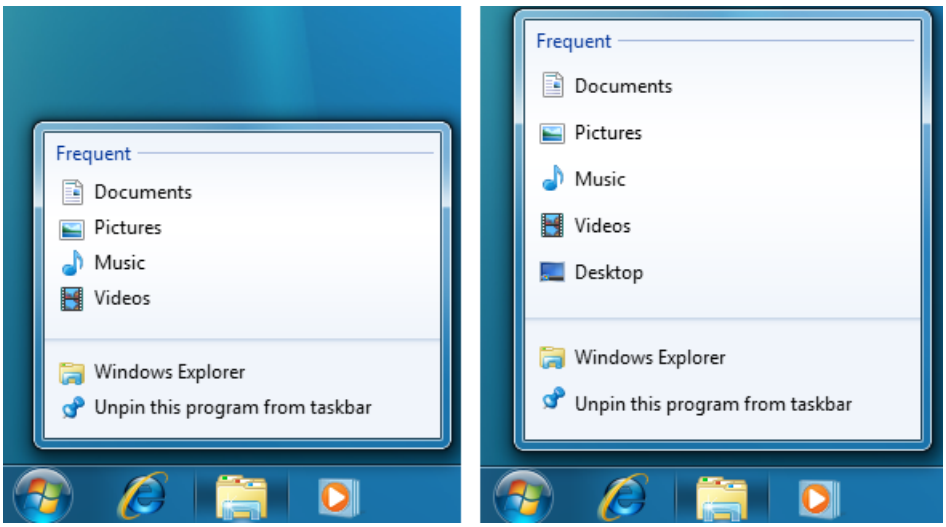

Abbildung 2.9: Die Listen der Taskleiste in Windows 7 haben größere Freiräume, wenn das System mit Berührungen gesteuert wird; Quelle: [11]

Auch die viel versprechenden Gesten könnten vor allem den unerfahrenen Benutzern einer Multitouch-fähigen Anwendung große Probleme bereiten. Einen der Vorteile von Gesten bildet die intuitive Steuerung. Doch was passiert, wenn dem Benutzer eine bestimmte Geste gänzlich unbekannt ist? Oder wenn eine Geste zu verspielt und kompliziert wirkt? Folglich muss sich der Benutzer Hilfe zu der Bedienung besorgen und die entsprechenden Gesten anlernen, bis er die Applikation korrekt bedienen kann. Solch ein Fall erschwert eine gute Benutzerfreundlichkeit und würde sogar eine Alternativbedienung erfordern. Allerdings sind Gesten bei der Verwendung von Multitouch-Geräten im heutigen Alltag allgemein noch relativ unbekannt, mit Ausnahme der Zoom-, Rotate-, Drag- und Tap-Gesten. Mit der zunehmenden Anteilnahme solcher Geräte im Alltag und

dem Festlegen neuer Gesten wird diesem Problem aber mit der Zeit entgegengewirkt.

Für großen Diskussionsstoff sorgten im Februar 2010 kritische Aussagen von einem Flash-Entwickler. So beschrieb Entwickler Daniel Eran Dilger seine Befürchtungen, dass derzeitige Flash-Inhalte niemals zufriedenstellend auf Touch-Geräten bedienbar sein werden: „[. . .] nothing can give users of any touchscreen, from any company, an acceptable experience with today's Flash sites. The thing so many complainers want is simply an impossibility“ [26]. Der Grund dafür ist, dass bei Eingaben auf Touchscreens der bei Flash oft verwendete Mouse-Over-Effekt fehlt. Damit werden ausfahrende Navigationsmenüs nutzlos. Eingeblendete Fenster mit Zusatzinformationen zu bestimmten Produkten, wenn man mit der Maus darüber fährt, können nicht mehr angezeigt werden. Hilfe versprach sich Dilger unter anderem von einer kompletten Touch-Erweiterung aller Flash-Projekte sowie von Implementierungen bestimmter Gesten, die den Mouse-Over-Effekt ersetzen. Jedoch wäre beides mit einem immensen Aufwand verbunden und auch nicht für jedes Projekt ohne Weiteres realisierbar. Durch den Einsatz von Multitouch-Komponenten würden diese Probleme innerhalb von Flash und Flex enstehen, auf welche innerhalb der Entwicklung geachtet werden muss.

Im Vordergrund der Multitouch-Technologie steht eine Bedienung auf der Grundlage mehrerer Berührungspunkte. Doch sollte dabei stets eine alternative Navigation mit einem Finger beziehungsweise einem Cursor gewährleistet werden. Dadurch würden sowohl die Gruppen der körperlich beeinträchtigten Menschen als auch Nutzer ohne Multitouch-unterstützende Hardware berücksichtigt werden. Es wäre unverantwortlich, eine Anwendung vollständig auf Multitouch auszurichten und somit eine Gleichberechtigung aller Nutzergruppen zu unterbinden

Mit dem Einsatz eines Touchscreens fehlt mit der nicht mehr vorhandenen Tastatur eine für den Benutzer bisher wichtige Eingabemöglichkeit. Für diese Komponente gibt es bislang meist einen Ersatz in Form einer virtuellen Tastatur. Diese muss sich den Platz auf dem Bildschirm mit der aktuellen Anwendung teilen. Das spart zwar Platz auf der Arbeitsfläche des Benutzers, allerdings werden damit auch einige Nachteile in Kauf genommen. Denn dem Benutzer wird das gewohnte taktile Gefühl für die Tasten genommen. Er verliert durch den fehlenden Widerstand beim Betätigen der virtuellen Tasten die Sicherheit am Schreiben. Dem Benutzer wird das Berühren einer Taste maximal durch eine grafische Hervorhebung der virtuellen Taste angezeigt. Darüber hinaus ist die Verbindung von Tastatur und Bildschirm nun eine andere. Bei der Eingabe mit einer physikalischen Tastatur kann ein geübter Benutzer ohne Blick auf die Tastatur die eingegebenen Buchstaben auf dem Bildschirm sehen. Wenn sich eine virtuelle Tastatur allerdings dauerhaft in dem Blickfeld des Nutzers befindet, ist dieses gewohnte Verhältnis gestört. Auch ist die körperliche Haltung bei der Verwendung der Tastatur eine andere. Die Relevanz dieser Problematik ist jedoch zu diesem Zeitpunkt nicht sicher und kann erst zu einem späteren Zeitpunkt untersucht werden. Nämlich dann, wenn die Touchfähigen Geräte und Anwendungen einen noch stärkeren Einfluss auf die verschiedenen

Nutzergruppen und deren Interaktionsverhalten ausüben.

2.2 Flex 4 Framework

Nach dieser Erklärung der Multitouch-Technologie werden im folgenden Abschnitt das Framework und die Entwicklungsumgebung für das Erstellen Multitouch-fähiger Flex-Komponenten erläutert. Bei Adobe Flex handelt es sich um ein englischsprachiges Framework, das unter der Open-Source-Lizenz steht. Haupteinsatzgebiet ist die Erstellung von Rich Internet Applications (RIA). Dabei handelt es sich um Anwendungen, die im Internet ohne vorherige Installation lauffähig sind und mit dem Nutzer interagieren können. In der aktuellen Version Adobe Flex 4 können Applikationen erstellt werden, die mit Hilfe des Adobe Flash Players im Browser oder mit Adobe AIR auf dem Desktop des Nutzers ausführbar sind. Wird eine Multitouch-fähige Flash-Anwendung in einem Browser verwendet, muss dieser das Flash Player Plug-In der Version 10.1 oder höher implementiert haben.

In dem Framework muss der Entwickler eine Programmier- und eine Auszeichnungssprache einsetzen. Dazu wird das auf Extensible Markup Language (XML) basierende MXML verwendet, um alle sichtbaren und nichtsichtbaren Komponenten einer Anwendung festzulegen. Action Script 3, welches ebenfalls in Adobe Flash Professional ab Version 9 eingesetzt wird, dient hingegen zur objektorientierten Implementierung der Funktionalität [22].

In der aktuellen Version des Frameworks werden einige Klassen zur Entwicklung von Multitouch-Anwendungen angeboten, die im Rahmen dieser Arbeit hohe Priorität besitzen. Neben der Möglichkeit, die Daten einzelner Touch-Events abzufragen und mit ihnen zu arbeiten, bietet Flex vereinzelte Funktionen zu Gestenerkennung an. Unterstütze Gesten sind beispielsweise das Zoomen und Rotieren von Objekten oder das gleichzeitige Auswählen mit zwei Fingern.

Mit seinen in Klassenbibliotheken festgelegten über 100 grafischen Elementen bietet das Framework dem Programmierer reichhaltige Möglichkeiten, Anwendungen aufzubauen und mit Programmierlogik auszustatten. Daher ist es für professionelle Entwicklungen von RIAs schon längst eine Alternative zu Adobe Flash Professional oder vergleichbarer Software. Im Rahmen der Arbeit wird innerhalb von Flex zu einem großen Teil mit den Komponenten aus dem »Halo«-Paket gearbeitet, die anhand des Namensraumes *MX* deklariert werden. Die Komponenten sind in Adobe Flex 4 weiterhin ohne Einschränkungen funktionsfähig und waren bis zur Flex-Version 3 aktuell. Mit der Einführung von Flex 4 wurden einige dieser Komponenten durch das aktuellere »Spark«-Paket ersetzt, welches den Namensraum *S* mit sich brachte.

2.2.1 Adobe AIR

Adobe AIR ist eine kostenlose und plattformunabhängige Laufzeitumgebung, die zur Erstellung von RIAs für den Desktopeinsatz verwendet wird. AIR-Anwendungen können in aktuellen Adobe Flash Professional-Versionen, in Flash Builder und in Adobe Dreamweaver entwickelt werden. Dazu ist unter Umständen eine Installation des kostenlosen AIR Software Development Kit (SDK) nötig.

Ab der Version 2.0 wurden Funktionen wie der Zugriff auf das lokale Dateisystem des Nutzers weiterentwickelt und neue, wie die Unterstützung von Multitouch, erstmals implementiert. Da es sich mittlerweile auch auf mobilen Endgeräten laufenden Betriebssystemen wie Android ausführen lässt, wurde die Plattformunabhängigkeit nochmals erweitert. Weiterhin heben sich AIR-Anwendungen durch die Implementierung umfangreicher Drag And Drop-Funktionen und dem Einsatz von Systemtastenkürzel von anderen Desktopanwendungen ab.

Adobe Flex und AIR stehen in einer engen Verbindung zueiander. Dabei fungiert AIR ähnlich wie der bekannte Flash Player als Laufzeitumgebung zum Ausführen der mit Flex erstellten Applikationen. Der Nutzer, der die kompilierte AIR-Datei ausführen möchte, benötigt dazu lediglich eine aktuelle Version der Runtime. AIR-Anwendungen können direkt im Internet aufgerufen und heruntergeladen werden, wodurch sie sich nach einer kurzen Installation ausführen lassen.

2.2.2 Entwicklungsumgebung

Als Entwicklungsumgebung für diese Arbeit wurde Flash Builder gewählt. Dieser verwendet unter Anderem das Flex 4 Framework und unterstützt die Erstellung von AIR 2.0-Anwendungen. Grundsätzlich ist der Flash Builder in seinen Versionen kostenpflichtig. Im Rahmen der Bachelorarbeit wurde mit der für Studenten kostenlosen Educational Version des Adobe Flash Builder Standard gearbeitet. Diese bringt nahezu alle Funktionen der Premium-Version mit sich, was eine Entwicklung von Multitouch-fähigen Komponenten gewährleistet.

Bei Flash Builder handelt es sich um ein Entwicklungswerkzeug, das auf der Eclipse Plattform basiert und mit dem sich sowohl RIAs als auch Desktopanwendungen entwickeln lassen. Wichtige Umgebungsbereiche sind der Editor und der Debugger, die in den letzten Versionen stetig weiterentwickelt wurden. Ein Vorteil des Editors ist das Umschalten zwischen Design- und Source-Modus. Das bedeutet, dass der Entwickler zwischen dem Quellcode und einer Vorschauansicht der grafischen Komponenten wechseln kann. Weiterhin wurden mit der aktuellen Version des Flash Builders auch die Möglichkeiten zum Datenaustausch beispielsweise mit Hilfe von Java oder Adobe Cold Fusion deutlich verbessert.

3 Analyse von Cyber

In diesem Kapitel wird eine bestehende Anwendung analysiert. Diese stellt im weiteren Verlauf der Arbeit die Grundlage für das Erstellen eines Konzeptes dar. Dabei werden zunächst sowohl das Front-End[12] als auch das Back-End[13] bezüglich ihres Aufbaus und der Funktionsweise untersucht. Weiterhin wird der Benutzerfreundlichkeit der Anwendung hohe Priorität zugeteilt. Im Anschluss wird die Anwendung auf einen ausgewählten Bereich beschränkt und dieser in all seinen Komponenten und Funktionen erläutert.

Die zu analysierende Applikation Cyber wurde von der Agentur queo GmbH entwickelt. Bei Cyber handelt es sich um eine Simulationsanwendung. Damit findet der Benutzer ein umfangreiches Werkzeug bei der Planung von Entwicklungs- und Produktionsabläufen eines Produktes. Als Anwendungsgebiet gilt hierfür die Fahrzeugproduktion. Mit Cyber lässt sich das Portfolio[14] des Auftraggebers in Form von Simulationsmodellen visualisieren [24]. Dabei werden Szenarien von unterschiedlichen Fahrzeugtypen oder Vergleiche zwischen verschiedenen Planungszeiträumen dargestellt. Änderungen in den Szenarien wirken sich umgehend auf das Portfolio aus. Weiterhin erstreckt sich der Funktionsumfang vom Erstellen bis hin zur Ausgabe von vorgefertigten Berichten mit Hilfe bestimmter Module. Als Front-End fungiert dabei eine Adobe Flex-Applikation, die ihre Daten aus einem auf Java basierendem Back-End bezieht.

3.1 Front-End

Das Front-End von Cyber und der damit für den Endbenutzer sichtbare Bereich der Anwendung stellt ein aus mehreren Klassen bestehendes Adobe Flex-Projekt dar. Mit Hilfe dieser Flex-Applikation lassen sich die über das Back-End erhaltenen Daten in vielerlei Hinsicht tabellarisch, grafisch oder anhand von Formularen darstellen.

3.1.1 Technologie

Die aktuelle Version von Cyber basiert auf einem Adobe Flex 4-Projekt. Dadurch wird das Arbeiten mit dem Flex 4-SDK gewährleistet. Ausgehend von der Main-MXML-Datei werden neben zahlreichen Komponenten aus den Flex Bibliotheken auch individuelle

[12] Im Beispiel der Cyber-Anwendung beinhaltet ein Front-End die für den Client sichtbaren grafischen Komponenten der Applikation.

[13] In Cyber wird ein Back-End als Schnittstelle zwischen Datenbank und der Anwendung eingesetzt. Alle dynamischen Datensätze werden darüber für die Einbettung in das Front-End generiert.

[14] Ein Portfolio beschreibt im informationstechnischen Sinn eine Sammlung von IT-Anwendungen und -Projekten, die beispielsweise als Referenz auf der Webseite des Unternehmens vorgestellt werden.

Komponenten in die Anwendung implementiert. Häufig liegen dabei erweiterte Klassen vor, die auf den vorgegebenen Flex-Komponenten aufbauen. Abbildung 3.1 verdeutlicht, wie dieses Verfahren beispielsweise bei dem individuell gestalteten Slider angewendet wird.

Abbildung 3.1: Links - eine unveränderte Flex-Slider-Komponente, rechts - der individuelle Slider in Cyber

Die über das Back-End erhaltenen Datensätze werden mittels Action Script-Klassen in neue Objekte zerlegt. Diese lassen sich dadurch als Datenprovider für grafische Komponenten nutzen. Mit den Komponenten kann der Benutzer agieren und zusätzlich Daten ändern. Die vom Nutzer getätigten Änderungen werden sofort an die entsprechenden Handler-Methoden weitergeleitet und über das Back-End in der Datenbank gespeichert.

Durch den modularen Aufbau des Projekts sowie der Einteilung der Action Script-Klassen und MXML-Komponenten in die entsprechenden Projektpakete können Änderungen an einzelnen Elementen mit geringem Zeitaufwand vorgenommen werden. Dabei ist es ebenfalls möglich, einzelne Komponenten außerhalb der Anwendung nach Anpassung einiger Beispieldatensätze für Testzwecke zu verwenden.

3.1.2 Aufbau der Benutzeroberfläche

Im Folgenden werden die Benutzeroberfläche und die damit zusammenhängenden grafischen Komponenten betrachtet. Dabei fällt vor allem die sogenannte *Detailansicht* als komplexer und umfangreicher Anwendungsbereich auf. Nach dem Starten der Applikation ist es dem Nutzer möglich, aus bereits vorhandenen Szenarien eine Auswahl zu treffen oder ein neues Szenario anzulegen. Wenn ein Szenario aufgerufen wurde, werden die entsprechenden Inhalte in einem neuen Fenster dargestellt. Dabei handelt es sich um die *Detailansicht*. In dieser Ansicht findet der Nutzer neben Filter- und Hilfefunktionen auch Übersichten, welche die Projektdaten visualisieren. Eine Übersicht der wichtigsten Elemente innerhalb der Detailansicht bildet die Abbildung 3.2. Dabei liegt bereits eine Einteilung der Oberfläche in folgende, wesentliche Bereiche vor: Filter, Auswahl, Selektion, Summe, Details, Navigation und Hilfe sowie der für die Arbeit vernachlässigbare Bereich Rückmeldung.

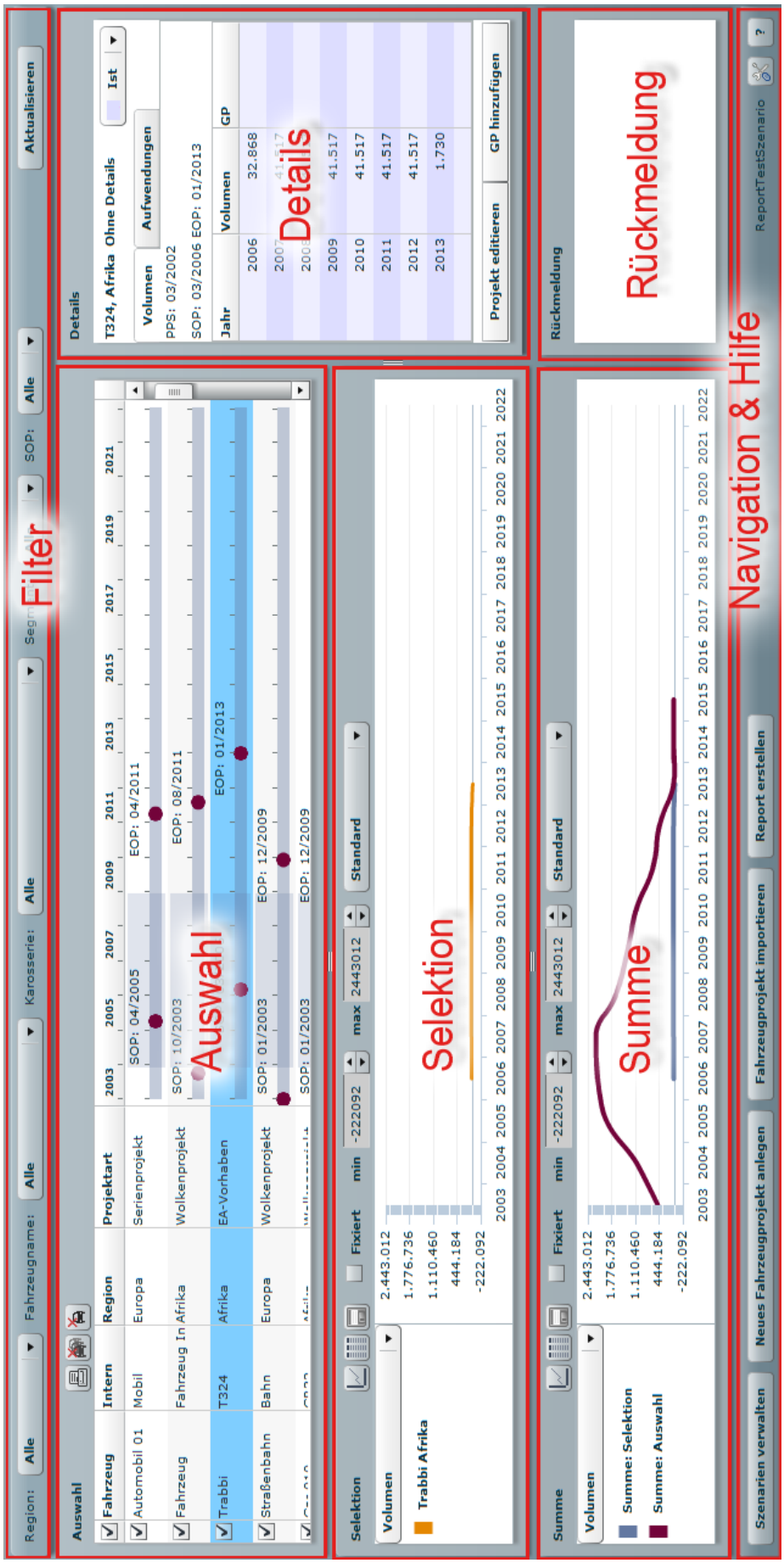

Abbildung 3.2: Die *Detailansicht* von Cyber mit Hervorhebung der wesentlichen Bereiche

Im oberen Bereich der *Detailansicht* wurde eine Menüleiste mit mehreren Filterfunktionen implementiert. Mit ihrer Hilfe kann nach Daten wie Fahrzeugnamen, Karosserie oder Region gefiltert werden. All dies wird anhand der in Abbildung 3.3 erkennbaren Drop-Down-Menüs realisiert. Die Form solcher horizontal angeordneten Menüs hebt sich von den restlichen Übersichten der Anwendung ab und erinnert an typische Navigationsmenüs aus dem Webdesign. Dem Benutzer wird dadurch signalisiert, dass er in diesem Bereich Einstellungen vornehmen kann, die sich auf die komplette Anwendung auswirken.

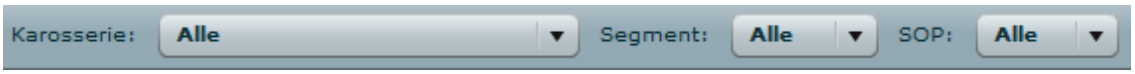

Abbildung 3.3: Einige Filterfunktionen im oberen Bereich der Anwendung

Im Mittelpunkt der Benutzeroberfläche nehmen vier Übersichten und ein Feld für zusätzliche Anweisungen einen Großteil des gesamten Platzes ein. Hauptfunktion der vier Übersichten *Auswahl*, *Selektion*, *Summe* und *Details* ist es, die gewählten Projektdaten auf verschiedene Weisen darzustellen und die Daten mit bestimmten Funktionen zu bearbeiten. Für einen Benutzer, dem die Anwendung unbekannt ist, wirkt die Oberfläche samt ihren Übersichten und Menüleisten überladen. Eine große Menge an Informationsdarstellungen teilt sich den knappen Platz der Oberfläche. Die Möglichkeit, den gesamten Bereich des Bildschirmes ausnutzen zu können, kann einen unerfahrenen Benutzer damit schnell überfordern. Andererseits wird dadurch eine bestmögliche Gegenüberstellung verschiedener Darstellungen erreicht.

Damit der Benutzer nach dem Öffnen der *Detailansicht* zu der Komponente gelenkt wird, die die meisten Interaktionsmöglichkeiten bietet, befindet sich die Übersicht *Auswahl* im linken oberen Bereich der Oberfläche. Dort wird dem Nutzer eine Auflistung aller Fahrzeugprojekte des Szenarios angezeigt. Die Projekte sind, wie in Abbildung 3.4 erkennbar, listenförmig untereinander angeordnet. Neben den Informationen zum Fahrzeug und der Projektart wird innerhalb der Tabelle auch ein Zeitstrahl abgebildet. In diesem werden Start of Production (SOP), End of Production (EOP) sowie optionale Große Produktaufwertungen (GP) als kreis- oder rautenförmige Grafiken visualisiert. Diese Informationen dienen dazu, einen Produktionsablauf in seine wesentlichen zeitlichen Abschnitte einzuteilen. Dabei steht der SOP für den Beginn eines solchen Ablaufes. Ein EOP kennzeichnet hingegen seinen Abschluss. Optionale Abschnitte oder Zwischenziele werden anhand der GPs markiert. Diese können einen Produktionsablauf in beliebiger Anzahl unterteilen, sind aber im Gegensatz zu SOP und EOP nicht zwingend notwendig.

Wird im Fenster *Auswahl* ein Fahrzeugprojekt angeklickt, stellt eine weitere Übersicht Informationen bezüglich dessen Volumen in einem Liniendiagramm dar. Diese Funktion wird dem *Selektion*-Fenster zugeschrieben und wird in Abbildung 3.5 beispielhaft dargestellt. Neben dem Volumen sind über ein Drop-Down-Menü auch Aufwendungen oder

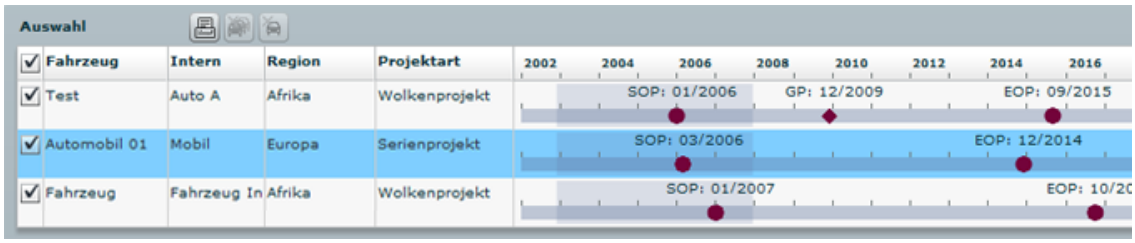

Abbildung 3.4: Ein Ausschnitt aus dem *Auswahl*-Fenster mit drei aufgelisteten Beispielprojekten

die Produktkapazität einstellbar. Ebenfalls in diesem Fenster befinden sich Schaltflächen zum Wechseln zwischen Diagrammansicht und Tabellenansicht sowie zum Export der Daten.

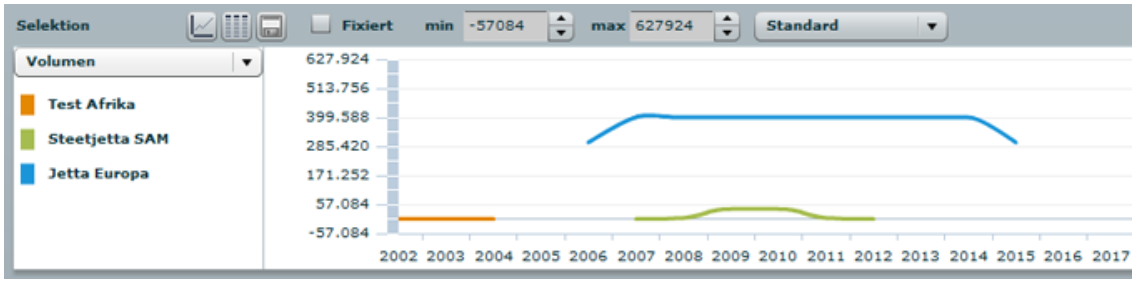

Abbildung 3.5: Das *Selektion*-Fenster mit allen Aufwendungen eines ausgewählten Beispielprojektes

Ähnlich wie das *Selektion*-Fenster arbeitet auch die Übersicht *Summe*. Diese stellt ebenfalls in einem Liniendiagramm Daten von Fahrzeugprojekten dar. Der Unterschied der beiden Übersichten besteht darin, dass im Gegensatz zur *Selektion* bei der Übersicht *Summe* sämtliche Fahrzeugprojekte innerhalb eines Szenarios einbezogen werden. Dadurch wird, wie in Abbildung 3.6 zu sehen, eine Vergleichsdarstellung ermöglicht. In seinen weiteren Funktionen ist dieses Fenster mit der *Selektion*-Übersicht hinsichtlich Darstellungsweisen und Berechnungsmethoden identisch.

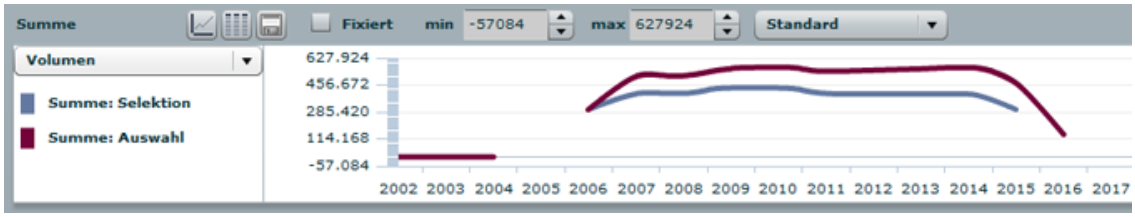

Abbildung 3.6: Das Fenster *Summe* vergleicht ausgewählte Projekte aus den Übersichten *Auswahl* und *Selektion*

Ist ein Fahrzeugprojekt im Fenster *Auswahl* aktiv, zeigt die Übersicht *Details*, wie in Abbildung 3.7 erkennbar, diesbezüglich nähere Informationen an. Dabei wird der Nutzer über Volumen und Aufwendungen des Fahrzeugprojektes informiert. Weiterhin können die Daten zu SOP, EOP, Berechnungsmethoden oder allen anfallenden Kosten eingesehen und in einer zusätzlichen Ansicht bearbeitet werden. Alle dort eingegebenen und

gespeicherten Daten wirken sich unmittelbar auf die Übersichten in der *Detailansicht* aus. Schaltflächen zum Aufruf der Cyber-Hilfe und dem Einstellen individueller Betrachtungszeiträume sowie Nutzerrechte befinden sich zusätzlich im unteren Bereich der Detailansicht.

Abbildung 3.7: Ausschnitt aus dem *Details*-Fenster, das nähere Informationen zu einem gewählten Projekt auflistet

3.1.3 Funktionsweise

Wie im vorigen Abschnitt erwähnt, arbeitet Cyber mit Szenarien. In einem Szenario können Fahrzeugprojekte angelegt und auf verschiedene Arten angezeigt werden. Dies ist eine grundlegende Funktion der Software. Denn durch die übersichtliche Darstellung von produktionsrelevanten Daten verschiedener Fahrzeuge können diese miteinander verglichen und Produktionsabläufe aufeinander abgestimmt werden. Die Darstellungen werden dabei anhand der Flex-Charting-Komponente realisiert. Änderungen der ursprünglichen Diagrammdaten wirken sich dadurch in Echtzeit auf das Diagramm aus. Zusätzlich bieten sowohl das *Selektion*- als auch das *Summe*-Fenster eine alternative Darstellung der Daten in Form einer Tabelle an. Dabei werden die relevanten Daten innerhalb des verlangten Zeitrahmens angezeigt und lassen kleinere Unterscheide besser ausmachen.

Neben diesen Visualierungen bietet Cyber eine Übersicht der zeitlichen Projektabschnitte in Form von SOP, EOP und GP. In der in Abbildung 3.8 sichtbaren Slider-Komponente sind diese auf einem Zeitstrahl angeordnet. Da jedes Fahrzeugprojekt mit einem eigenem Zeitstrahl in einer gemeinsamen Tabelle eingebunden ist, bekommt der Nutzer einen besseren Überblick über aktuelle und zukünftige Produktionen sowie parallele Abläufe.

Um ein Szenario auch außerhalb von Cyber zu verwenden, bietet die Anwendung eine Auswahl an Exportmöglichkeiten. Im Fenster *Auswahl* wird dem Nutzer angeboten, die dort dargestellte Übersicht als PDF-Dokument zu exportieren. In dem generierten Dokument finden sich alle zum Szenario gehörigen Fahrzeugprojekte inklusive Fahrzeuginformationen und Produktionsdaten wieder. Darüber hinaus bietet Cyber die Mög-

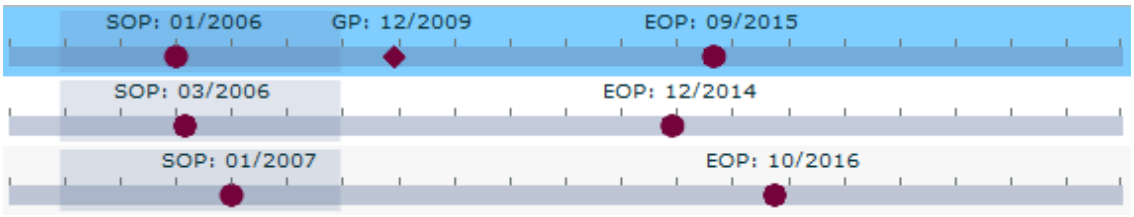

Abbildung 3.8: Der Zeitstrahl dreier Projekte wird anhand der Flex-Slider-Komponente realisiert

lichkeit, die in den Übersichten dargestellten Ergebnisrechnungen als XLS[15]-Datei zu exportieren oder einen Report zum aktuellen Szenario zu erstellen und anzuwenden.

Mit diesen Funktionen ist Cyber eine effektive und umfangreiche Applikation, die komplexe Abläufe und Daten auf übersichtliche Art und Weise darstellt. Mit seinen Möglichkeiten zum Export von Übersichten und Reporten können die bei der Arbeit mit Cyber erstellten Szenarien in anderen Unternehmensbereichen und Anwendungen verwendet werden.

3.1.4 Bedienung der Anwendung

Um die Bedienung von Cyber genauer zu untersuchen, muss vorrangig die *Detailansicht* analysiert werden. Diese bietet dem Benutzer bereits aus dem Bereich der Web-Applikationen bekannte Interaktionsmöglichkeiten.

Im Fenster *Auswahl* finden sich vertraute Komponenten wie Scroll-Leisten, Drop-Down-Menüs oder Checkboxen wieder, deren korrekte Handhabung man von einem Großteil der Besucher von Webseiten voraussetzen kann. Diese Komponenten können ohne spezielle Vorkenntnisse mit der Maus bedient werden. Im Gegensatz dazu wird die Tabellen-Komponente für unerfahrene Benutzer als neu eingestuft. Es wird zwar ersichtlich, dass die Fahrzeugprojekte in einer Liste angeordnet werden. Dass diese aber mit Hilfe der Hauptfunktion ausgewählt und die damit verbundenen Informationen übergreifend gepflegt werden, erfordert eine gewisse Einarbeitungszeit. Tabellen können innerhalb von Adobe Flex als nützliche Werkzeuge eingesetzt werden und sind vielen Flash- und Flex-Benutzern vertraut. Geraten jedoch unerfahrene Personen erstmalig in Kontakt mit diesen interaktiven Komponenten, erwarten sie nicht, dass einzelne Listenelemente anklickbar sind. Die Tabelle in Form einer Flex-DataGrid ermöglicht dem erfahrenen Benutzer noch weitere Interaktionsmöglichkeiten. Dabei lassen sich standardmäßig die Tabellenspalten bei gedrückter linker Maustaste verschieben. Zudem können die einzelnen Grafiken der Produktionsabschnitte mit einer Drag and Drop-Bewegung im Zeitstrahl versetzt werden. Beides sind Verhalten, die der Benutzer nicht erwartet, welche aber die Verwendung der Applikation vereinfachen können.

[15] In Microsoft Excel erstellte Projekte liegt meist im XLS (Excel Spreadsheet)-Format vor.

Die Benutzerfreundlichkeit oder auch Usability der Anwendung soll auch bei umfangreichen Funktionalitäten gewährleistet sein. „Mehr Funktionalität heißt [...] nicht immer mehr Usability. Eher das Gegenteil ist der Fall!“ beschreiben Werner Schweibenz und Frank Thissen in ihrem Buch über das benutzerfreundliche Erstellen von Webseiten [32]. Ähnliches trifft auch auf die Funktionen der Tabelle zu, die mit der Tastatur ausführbar sind. Der Einsatz einer Tastatur innerhalb von Web-Aplikationen bringt nicht automatisch Nachteile mit sich. Im Fall von Cyber können Funktionen der Tabelle mit drei verschiedenen Tasten aufgerufen werden. Dabei wird dem unerfahrenen Benutzer allerdings nicht ohne Blick in die Hilfe ersichtlich, mit welcher Taste eine spezielle Funktionalität angesprochen wird. So sind Tabellenzeilen mit Halten der *<STRG>*-Taste an- und abwählbar. Auch dieses Verhalten kann der Benutzer aus anderen Anwendungen kennen, in denen Objekte zu einer Auswahl durch Festhalten der *<STRG>*-Taste hinzugefügt werden. Doch durch das Vorhandensein der Checkboxen in der Tabelle wird eine Auswahl-Bearbeitung mit Hilfe der Checkboxen als selbstverständlich interpretiert. Da Checkboxen aus zahlreichen Web-Anwendungen bekannt sind, werden sie als beliebtes Mittel zur An- und Abwahl eines Listenelementes eingesetzt. Cyber unterstützt jedoch beide Auswahlmöglichkeiten. Erst nach intensiver Bearbeitungszeit mit der Software wird der Benutzer die Unterschiede der Auswahlfunktionen erkennen. Über die Checkboxen sind die Daten für das Diagramm in dem *Summe*-Fenster einstellbar. Unabhängig von den Checkboxen kann der Benutzer die Projektdaten für die Übersicht *Selektion* anhand der Tabellenzeilen bearbeiten. Beides sind einfache und zuverlässige Methoden, um Projekte aus der Tabelle für die jeweilige Darstellung auszuwählen. Doch bis der Benutzer den Unterschied zwischen ihnen erkannt hat, muss er sich erst genauer mit der Tabelle beschäftigen.

Weiterhin bekommt der Benutzer die Möglichkeit, die Grafiken des Zeitstrahls zu bewegen. Neben dem einfachen Drag and Drop eines einzelnen Objektes, können auch alle Slider-Grafiken in einer Zeile auf einmal verschoben werden. Dieses Wissen muss sich der Benutzer aus der Hilfe aneignen. Mit gehaltener *<SHIFT>*-Taste muss ein SOP oder EOP mit der linken Maustaste ausgewählt werden. Anschließend sind die Elemente verschiebbar und orientieren sich an der Position des Cursors. Die *<ALT>*-Taste bewirkt eine weitere Funktion. Soll eine neue GP in dem Zeitstrahl hinzugefügt werden, braucht der Benutzer die Taste zu halten und auf die Stelle zu klicken, an der die GP erscheinen soll. Eine ähnlich einfache Funktion zum Löschen gibt es nicht. Löschen kann der Benutzer eine GP nur über das Menü zum Editieren der Projektinformationen. Diese Möglichkeiten, eine Funktionalität mit gehaltener Taste aufzurufen, haben natürlich den großen Vorteil, dass der Benutzer sich nicht erst duch entsprechende Menüs arbeiten muss. Allerdings bleibt Benutzern ohne Erfahrung mit Cyber die Existenz dieser Möglichkeiten ohne Blick in die Hilfe verborgen.

Die beiden Fenster *Summe* und *Selektion* sind im Gegensatz zu dem *Auswahl*-Fenster einfacher bedienbar, haben aber auch andere Funktionalitäten. Hierbei regelt man alle Einstellungen über Drop-Down-Menüs und kann über Buttons die Ansichten wechseln

sowie die Übersicht exportieren. Funktionen zum Umschalten der Ansichten werden mit kleinen, selbsterklärenden Symbolen gekennzeichnet. Dies ist eine Möglichkeit, dem Benutzer den Zugriff auf wichtige Funktionen zu gewährleisten. Damit wird auch Anwendern, denen die Anwendung bislang unbekannt ist, kurz aufgezeigt, welche Funktionalitäten sich hinter den Schaltflächen verbirgt.

Zur besseren Ansicht der einzelnen Fenster in der *Detailansicht* gibt es zwischen den Fenstern Divider-Komponenten, mit deren Hilfe sich Fenstergrößen nach Belieben anpassen lassen. Wie in Abbildung 3.9 verdeutlicht, kann der Divider durch Ziehen mit der linken Maustaste verschoben werden. Wird die Maustaste an der gewünschten Stelle losgelassen, nimmt der Divider diese Position ein und die daran angrenzenden Panels richten ihre Größe danach aus. Dadurch wird dem Nutzer gewährt, auf unterschiedliche Szenarien einzugehen und die gute Übersicht stets zu wahren. Außerdem wirkt die Anwendung weniger statisch. Die restlichen Funktionen in der *Detailansicht* werden über weitere Buttons und Drop-Down-Menüs geregelt.

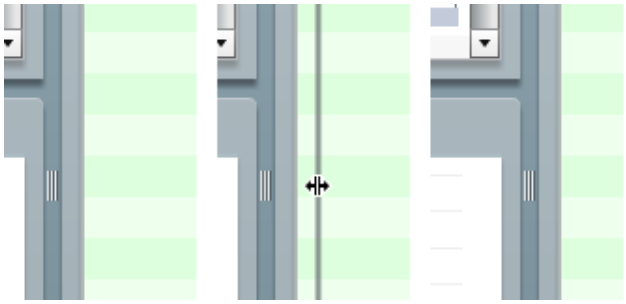

Abbildung 3.9: Durch Ziehen und Loslassen des horizontalen Dividers lässt sich die Panelgröße ändern

Viele weitere Einstellungen werden in aufspringenden Fenstern durch die Flex-Panel-Komponente dargestellt. Dort hat der Benutzer meist zahlreiche Möglichkeiten, mit Hilfe von Buttons, Textfeldern oder Drop-Down-Menüs Daten einzusehen und zu bearbeiten. Ein Beispiel dieser Eingabeformulare stellt das in Abbildung 3.10 sichtbare *Regionen*-Fenster dar.

Abbildung 3.10: Eines der zahlreichen Menüs zum Editieren der Datensätze - hier am Beispiel der *Regionen*-Eigenschaft

Mit diesen Formularen geht aber auch eine benutzfreundliche Übersicht verloren. Durch die Überlagerung mehrerer solcher Pop-Up-Fenster liegen oftmals zwei Formulare über der eigentlichen Anwendung, die noch leicht hindurchleuchtet. Besser wäre ein getrennter Bearbeitungsmodus, der sich klar von der grafischen Übersicht aus der *Detailansicht* abhebt. Damit würde auch der Benutzer eine strikte Grenze zwischen Bearbeitung der Daten und grafische Darstellung vorfinden. Außerdem ist es möglich, dass man bei der ersten Einsicht in die Bearbeitungs-Menüs nicht sofort die gewünschten Einstellungen findet. Die Abbildung 3.11 zeigt, dass ein optisch gleichwirkendes und übersichtliches Einbetten der Schaltflächen in das GUI nicht gegeben ist. Große Freiräume in der Oberfläche bewirken, dass das Menü unausgewogen wahrgenommen wird.

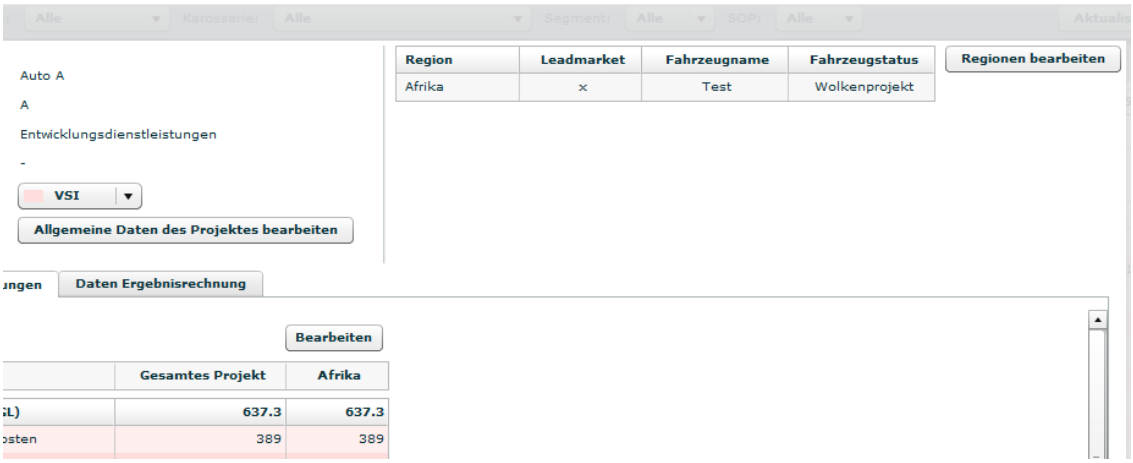

Abbildung 3.11: Ein Ausschnitt aus dem Menü zum Editieren der Projekteigenschaften

Ein Großteil der soeben untersuchten Bedienelemente gibt dem Benutzer die Möglichkeit, ohne viel Arbeitsaufwand Änderungen an den Projektdaten vorzunehmen oder die Darstellungsformen nach Belieben anzupassen. Funktionen, die nur in Verbindung mit einer gedrückten Taste aufrufbar sind, können sich jedoch zumindest für Anwender, denen Cyber unbekannt ist, als problematisch erweisen. Außerdem bewirkt die teils unübersichtliche Darstellung von Formularen eine gewisse Orientierungslosigkeit, wenn eine spezielle Einstellung vorgenommen werden soll. Erfahrene Benutzer hingegen können schnell mit interaktiven Elementen wie der Tabelle oder dem Slider die notwendigen Einstellungen vornehmen, womit die Verwendung der Formulare gemieden werden kann.

3.2 Back-End

Mit dem Back-End der Cyber-Applikation wird die Kommunikation zwischen dem bereits beschriebenen Front-End und einer Datenbank gewährleistet. Im Mittelpunkt stehen dabei zahlreiche Java-Klassen, welche auf die gestellten Anfragen aus dem Front-End reagieren.

3.2.1 Technologie

Das Back-End von Cyber läuft in einem Java Servlet Container. In der Entwicklungsumgebung wird dieser anhand eines Apache Tomcat[16] realisiert. Bei der Verwendung in einer Produktiv- oder Testumgebung wird hingegen ein IBM WebSphere Application Server eingesetzt. Dieser Server dient als Laufzeitanwendung für Java Enterprise Edition-Anwendungen. Wie im Fall von Cyber können diese Anwendungen als Web Application Archive (WAR)[17] vorliegen und werden im Applikationsserver entpackt.

Der Datenaustausch des Back-Ends beruht auf der Kommunikation zwischen Front-End und Datenbank per XML. Die Datensätze, die an das Front-End weitergegeben werden, befinden sich in einer DB2-Datenbank[18] von IBM. Dort wurden Tabellen mit allen nötigen Informationen angelegt, die bei dem Einsatz von Cyber eine Rolle spielen.

Die Daten werden während des Kommunikationsvorganges im XML-Format an das Front-End übergeben. Ein Vorteil dieses Formates ist die einfache Einbindung der Datensätze in das Flex-Projekt sowie dem damit verbundenen Anlegen und Füllen von Action Script-Objekten. Auf der Serverseite findet ein XML-Object Mapping mittels *XStream* statt. Bei dieser Mapping-Variante können XML-Dokumente in Java-Objekte und umgekehrt Java-Objekte in XML-Dokumente umgewandelt werden [28]. *XStream* dient dabei als Mapping-Framework.

In dem Back-End werden mit Hilfe von *Spring 2* und *Hibernate 3* zwei wesentliche Kerntechnologien eingesetzt. *Spring 2* ist ein Framework, welches als Inversion of Control (IOC)-Container Verwendung findet. IOC ist ein spezielles Prinzip der Softwareentwicklung. Dabei erhält das Framework die Kontrolle über den Programmcode [14]. *Hibernate 3* hingegen stellt ein Object-Relational Mapping (ORM)-Framework für Java dar. Damit können Objekte mit Methoden und Attributen in relationalen Datenbanken gespeichert und entgegengesetzt aus den Datenbanken Objekte erzeugt werden [21]. Mit Mechanismen zum Zugriff auf Datenbanken ohne der explizierten Programmierung in SQL bleibt die Applikation von der Datenbank unabhängig.

3.2.2 Aufbau

Das auf dem Webserver befindliche Back-End ist in drei Bereiche gegliedert. Einer dieser Bereiche ist der Frontend Controller. Dessen Funktion liegt im Empfangen, Senden und Mappen der XML-Daten. Den zweiten Bereich stellt die Business-Schicht dar. In ihr befindet sich die Programmlogik, in der die bereitgestellten Daten verarbeitet werden. Mit der sogenannten Data Access Object (DAO)-Schicht wird das Back-End komplettiert. Das Entwurfsmuster DAO ist mit einer separaten Software-Schicht vergleichbar.

[16] Der Apache Tomcat wird als Umgebung für ausführbaren Java-Code auf Webservern eingesetzt.
[17] Ein WAR-Archiv ist ein Dateiformat zur Beschreibung der Verpackung einer Web-Applikation
[18] Eine DB2 entspricht einem relationalen Datenbankmanagementsystem von IBM.

Es wird beim Zugriff auf die Datenbank eingesetzt und bringt große Vorteile hinsichtlich der guten Wiederverwendbarkeit des Programmcodes sowie der Abstraktion des Datenzugriffes mit sich [2].

3.2.3 Funktionsweise

Das Back-End fungiert als Verbindung zwischen Front-End und der Datenbank. Bei Änderungen im Front-End durch den Benutzer werden aus dem Front-End heraus die entsprechenden Java-Anfragen aufgerufen und die vorgenommenen Modifikationen als Variablen übergeben. Nach erfolgreicher Einspeisung der Änderungen in die Datenbank erhält das Front-End aus dem Back-End eine Antwort mit den nun neu in der Datenbank eingetragenen Werten. Anschließend werden die Daten im Front-End nochmals ausgewertet und auf die entsprechenden Objekte verteilt. Abschließend werden die aktualisierten Objekte wieder den grafischen Komponenten zugewiesen. Der Benutzer nimmt die Vorgänge nur als kurze Aktualisierung der betreffenden Komponenten wahr.

Durch seine Robustheit und den vielen, auf die Aktionen des Benutzers eingestellten individuellen Klassen ist das Back-End bestens für einen Einsatz mit großen Datenmengen konzipiert. Klassen zur Ausnahme- und Fehlerbehandlung helfen darüber hinaus, mit der kompletten Anwendung störungsfrei zu arbeiten.

3.3 Beschränkung der Anwendung

Als Ausgangspunkt für die weitere Bearbeitung an diesem Thema wird die Bachelorarbeit auf einen bestimmten Teil von Cyber begrenzt. Dabei handelt es sich um das Fenster *Auswahl* aus der *Detailansicht*. Diese Übersicht bietet zahlreiche Funktionalitäten und gehört damit zu den umfangreichsten Komponenten der gesamten Applikation.

3.3.1 Funktionalität des ausgewählten Bereiches

Wie bereits im Abschnitt 3.1.2 *Aufbau* erläutert, dient das Fenster zum Anzeigen und Auswählen der Fahrzeugprojekte eines Szenarios. Damit steht es immer in unmittelbarem Kontakt zu den darunterliegenden Fenstern *Selektion* und *Summe*. Diese stellen die gewünschten Daten des ausgewählten Projektes grafisch oder in Tabellenform dar.

Außerdem kann der Inhalt des *Auswahl*-Fensters über verschiedene andere Fenster beeinflusst werden. So ist es möglich, über die Funktion *Projekt editieren* die wesentlichen Daten, die in der Tabelle des *Auswahl*-Fensters angezeigt werden, zu bearbeiten. Vorgenommene Änderungen wirken sich direkt auf die Tabelle aus. Über Buttons wie

Fahrzeugprojekt importieren und *GP hinzufügen* sind darüber hinaus neue Daten zur Tabelle hinzufügbar.

Doch auch im *Auswahl*-Fenster lassen sich Fahrzeugprojekte manipulieren. Die Daten zu SOP, EOP und GP sind mittels Verschiebung entlang des Zeitstrahls stets änderbar. Des Weiteren bietet diese Übersicht Funktionalitäten zum Löschen von Fahrzeugprojekten und Fahrzeugen sowie zum Export der aktuellen Übersicht. All jene Funktionen lassen dieses Fenster zu einem wichtigen Baustein sowie Dreh- und Angelpunkt in der Cyber-Anwendung werden.

3.3.2 Komponenten in der Benutzeroberfläche

Das Fenster baut auf einer Panel-Komponente aus dem Flex-Framework auf. Darauf sind alle sichtbaren und nichtsichtbaren Komponenten platziert, mit denen der Benutzer interagieren kann oder die im Hintergrund ohne Sichtbarkeit für den Benutzer ihre jeweilige Funktion erfüllen. Das Panel ist mit Hilfe von an der Seite angebrachten Dividern in seiner Höhe und Breite individuell veränderbar.

In der Titelleiste des Fensters sind nebeneinander drei Buttons platziert. Mit ihnen lassen sich Fahrzeugprojekte und Fahrzeuge löschen sowie die aktuelle Tabellenansicht exportieren. Abbildung 3.12 zeigt, dass die Funktionen anhand von kleinen Symbolen auf den Buttons verdeutlicht werden.

Abbildung 3.12: Drei Schaltflächen zum Export der Ansicht sowie zum Löschen von Projekten und Fahrzeugen

Den eigentlichen Hauptteil des Fensters macht eine Tabelle mit sechs Spalten aus. Davon haben vier Spalten die Funktion, die projektspezifischen Informationen zu Fahrzeug, Region, internem Projektnamen und Projektart anzuzeigen. Als Darstellungsform wurden dabei, wie in Abbildung 3.13 zu sehen, Textfelder gewählt.

Fahrzeug	Intern	Region	Projektart
Test	Auto A	Afrika	Wolkenprojekt
Automobil 01	Mobil	Europa	Serienprojekt

Abbildung 3.13: Die Tabelle mit den vier Spalten, die projektbezogene Daten in Textfeldern anzeigen

Die vorderste Spalte hingegen dient zur Auswahl von Projekten für die Vergleichsdarstellung im Fenster *Summe*. Dies wird durch die in Abbildung 3.14 gezeigten Checkboxen realisiert. Je mehr Checkboxen aktiv sind, umso mehr Fahrzeugprojekte werden

auch miteinander verglichen und deren Durchschnittswert im Diagramm des *Summe*-Fensters dargestellt. Mit der Checkbox in der Überschriftenzeile sind alle Projekte auf einmal an- und abwählbar.

Abbildung 3.14: Die erste Spalte der Tabelle bietet Checkboxen zur Manipulation der Anzeige im *Summe*-Fenster

Die letzte Spalte der Tabelle kommt einem großen Zeitstrahl gleich. In der für diese Anwendung erweiterten Slider-Komponente werden die verschiedenen Produktionspunkte jedes Fahrzeugprojektes dargestellt. In der obersten Zeile der Tabelle finden sich die Jahreszahlen von 2002 bis 2020 im Abstand von zwei Jahren wieder. Die Zeilen darunter bestehen aus je einer blauen Linie mit Markierungen für jedes Jahr. Solche Linien werden in Adobe Flex auch als Track eines Sliders bezeichnet. Auf diesem Track sind die Slider-Objekte verschiebbar. Mit Hilfe der Markierungen in jeder Zeile der Tabelle ist ersichtlich, für welches Zeitintervall Planungspunkte vorgesehen sind. Abbildung 3.15 zeigt, dass sich auf dem Track zudem rote Kreis-Grafiken für einen SOP und EOP sowie nach Bedarf rote Rauten für eine GP befinden. Durch Drücken und Ziehen der linken Maustaste sind diese Objekte horizontal verschiebbar. Ebenfalls können alle Punkte einer Zeile auf einmal mit gehaltener *<SHIFT>*-Taste komplett verschoben werden. Als weitere Funktion ist es möglich, eine neue GP bei gedrückter *<ALT>*-Taste dem Fahrzeugprojekt hinzuzufügen. Diese lässt sich danach ebenso mit gedrückter Maustaste entlang des Zeitstrahls verschieben.

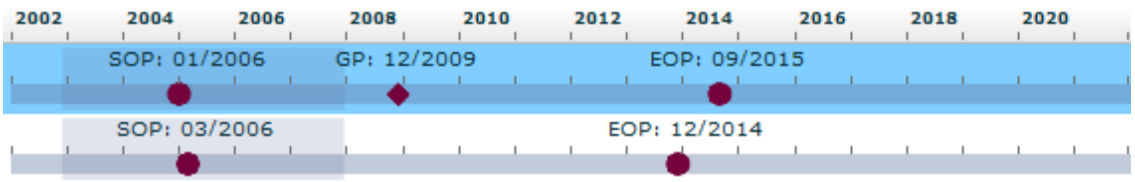

Abbildung 3.15: Der Zeitstrahl mit Slider-Komponente in den Tabellenzeilen und einer Intervallanzeige im Header

Über jedem SOP, EOP und GP werden in einer Label-Komponente Hinweise zu jedem Punkt angezeigt. Innerhalb von Adobe Flex werden solche individuellen Punkte auch als Thumb eines Sliders bezeichnet. Mit den Labels wird angegeben, um was für ein Ereignis es sich dabei handelt und in welchem Monat und Jahr es stattfindet. Wie in Abbildung 3.16 dargestellt, ändert sich nach der Verschiebung eines Thumbs der Text des Labels umgehend.

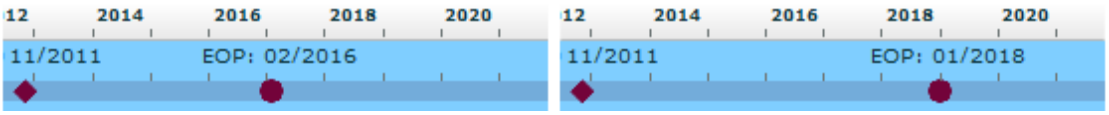

Abbildung 3.16: Das Label eines Slider-Thumbs ändert sich mit dessen Verschiebung

3.3.3 Auswahlbegründung

Da im Rahmen dieser Bachelorarbeit Multitouch-Komponenten für die Cyber-Anwendung entstehen sollen, musste sich auf einen Teil der Anwendung für die weitere Bearbeitung festgelegt werden. Die komplette Anwendung auf eine Steuerung mit Multitouch-Geräten einzustellen, würde diesen Rahmen weit übersteigen. Für die Entscheidung, das *Auswahl*-Fenster für weitere Bearbeitungschritte der Arbeit zu bestimmen, gab es verschiedene Gründe.

Einserseits gibt es in diesem Fenster verschiedene Interaktionsmöglichkeiten. Seien es Checkboxen, die Verschiebung der Produktionspunkte im Zeitstrahl unter Verwendung der Tastatur oder das Scrollen durch die Tabelle bei einer hohen Anzahl an Fahrzeugprojekten. All diese Funktionen bieten abwechslungsreiche Bedienformen und sollten im Laufe der Arbeit auf eine Verwendung mit einem Multitouch-Gerät angepasst oder gänzlich neu konzipiert werden. Die wesentlichen Hauptfunktionen des *Auswahl*-Fensters bleiben dabei erhalten.

Ein weiterer Punkt, der für die Auswahl des gewählten Fensters sprach, war die bisher nur wenig intuitive Bedienung. Der Einsatz einer Tastatur überfordert einen Nutzer ohne Vorkenntnisse bei der Bedienung der Software. Nur ein Blick in die Hilfe löst das Problem der korrekten Interaktion. Sichtlich anders wäre der Sachverhalt, wenn ein unerfahrener Benutzer intuitiv mit mehreren Fingern beispielsweise die kompletten Produktionspunkte verschieben könnte.

Des Weiteren ist die Größe des *Auswahl*-Fensters nicht immer an die aktuelle Menge an Fahrzeugdaten angepasst. In bestimmten Szenarien ist es durchaus möglich, dass eine große Anzahl an Fahrzeugprojekten die Tabellenzeilen ausfüllen und der Scrollaufwand erhöht wird. Damit geht auch eine benutzerfreundliche Übersicht verloren. In Cyber benutzt der Nutzer dafür die Dividier zum Ändern der Höhe und Breite des Fensters. Doch mit ihnen kann nur entweder die Breite oder die Höhe einer Übersicht mit einer Bewegung angepasst werden. Mit einer Aktion die komplette Größe eines Fensters zu ändern ist nicht möglich.

Diese aufgeführten Probleme müssen für die Arbeit an Cyber aufgegriffen werden, damit nicht nur die Verwendung mit einem Multitouch-Gerät ermöglicht wird, sondern sich auch die Bedienung intuitiver gestaltet. Es gibt in der gesamten Anwendung auch andere Bereiche, die darauf ausgelegt werden können. Dabei könnten zum Beispiel Lö-

sungsansätze für das Problem des Roll-Over-Effektes bei der Arbeit mit Drop-Down-Listen erarbeitet werden. Jedoch bietet das *Auswahl*-Fenster mit seinen verschiedenen Komponenten und Funktionalitäten das größte Potenzial für die Konzeption und Erstellung von Multitouch-Komponenten. Fertige Komponenten sollen in Weiterentwicklungen, die über diese Bachelorarbeit hinaus gehen, auch auf andere Bereiche von Cyber angewendet werden, die später ebenfalls bei der Verwendung von Multitouch-Geräten zu einem Einsatz kommen.

3.3.4 Konflikte bei der Verwendung auf Touchscreens

Im folgenden Abschnitt wird geprüft, welche bisherigen Interaktionsmöglichkeiten für eine Umstellung auf ein Multitouch-Bedienkonzept überarbeitet werden müssen. Da Touchscreens ohne Tastaturen bedient werden können, müssen jene Funktionalitäten, die nur unter Einsatz einer gedrückten Taste wirken, neu definiert werden. Das betrifft sowohl das gleichzeitige Verschieben aller Punkte einer Zeile im Zeitstrahl als auch das Hinzufügen einer neuen GP. Genauso muss auch die multiple Selektion von Projekten durch bisheriges Halten der *<STRG>*-Taste ersetzt werden. Für die aufgezählten Funktionen sind sowohl der Einsatz bestimmter Gesten als auch die Verwendung aufblendender und kontextbezogener Befehlsfenster möglich.

Wie in Kapitel 2.1.5 im Zusammenhang mit Problemen bei Multitouch-Anwendungen bereits erwähnt wurde, können unzureichende Oberflächendesigns zu massiven Problemen bei der Bedienung führen. Daher muss darauf geachtet werden, dass Elemente wie Checkboxen oder Buttons in der Cyber-Anwendung an eine Bedienung mit dem Finger angepasst werden. Die Schaltflächen liegen in Cyber in einer Größe von 22 mal 22 Pixeln vor. Diese Fläche ist mit einem Finger nur schwer anwählbar und erfordert eine Neugestaltung der Komponentengrößen. Ebenso müssen die im Zeitstrahl verwendeten Komponenten hinsichtlich ihrer Größe und Positionierung überdacht werden. Wenn ein Finger einen Slider-Thumb bewegt, darf er damit nicht die Zusatzinformationen verdecken, die über dem Punkt angezeigt werden. Neben der Anpassung der Objektgrößen ist daher auch eine Zoom-Funktion in die die Tabelle denkbar. Dadurch ließ sich ein zeitliches Intervall einstellen und die Abstände zwischen den Punkten vergrößern oder verkleinern, was eine bessere Ausgangslage zur Auswahl eines Punktes zur Folge hat.

Weiterhin wird aus dem bisherigen Oberflächendesign nicht ersichtlich, welche Elemente berührbar sind. Zweidimensionale Grafiken wie sie in der Slider-Komponente vorzufinden sind, wirken wenig plastisch. Lediglich die standardmäßigen Flex-Buttons zeigen dem Benutzer, dass diese Elemente mit einem Finger ausgewählt werden können. Jedoch ist auch die erwartete visuelle Reaktion der Anwendung auf eventuelle Berührungen sehr gering. Nur die Tabelle reagiert mit Farbänderungen einer selektierten Zeile auf Interaktionen mit dem Benutzer. Damit wird eine enge Verbindung zwischen Benut-

zer und Anwendung gestört.

In der Tabelle muss zusätzlich darauf geachtet werden, dass die Bedienung dieser Komponente später ohne den gewohnten Mouse-Over-Effekt auskommen wird. Bislang konnte sich der Benutzer über die Zeilen der Tabelle mit seinem Cursor bewegen und dadurch das Listenelement, über dem sich der Cursor befand, hellblau hinterlegen lassen. Auf Touchscreens fällt der Mouse-Over-Effekt aber weg, da der Mauszeiger durch Berührungspunkte ersetzt wird. Das heißt in der Cyber-Anwendung, dass die Auswahl einer Zeile nicht dann passieren darf, wenn der Benutzer mit seinem Finger über eine Zeile fährt. Dies hätte den Nachteil, dass ständig bei Berührung der DataGrid-Komponente und ihrer Kind-Elemente die Funktion zur Zeilenauswahl aufgerufen würde. Vielmehr ist es ratsam, eine Auswahl nur dann zu betätigen, wenn der Benutzer die gewünschte Zeile kurz angetippt hat.

3.4 Fazit

In dem Kapitel *Analyse* wurde ersichtlich, welche Funktionen die Simulationsanwendung Cyber mit all ihren Komponenten mit sich bringt. Mit der Möglichkeit, umfassende Datensätze auf verschiedene Weisen darzustellen und miteinander zu vergleichen, bekommt der Nutzer eine umfangreiche Applikation zur Verfügung gestellt. Dies gelingt sowohl durch Tabellen- als auch Diagrammdarstellungen, auf deren anzuzeigenden Daten der Nutzer einwirken kann.

Aus den Bereichen von Cyber wurde das Element *Auswahl*-Fenster zur weiteren Bearbeitung für die Bachelorarbeit ausgewählt. Diese Komponente wird zur tabellarischen Auflistung aller einem Szenario zugewiesenen Fahrzeugprojekte genutzt. Dabei baut sie sich sowohl aus standardmäßigen Flex-Komponenten als auch aus individuellen Elementen auf. Um dieses Fenster korrekt bedienen zu können, muss bei bestimmten Funktionen die Tastatur zu Hilfe genommen werden. Dieser Umstand erfordert eine Neugestaltung der bisherigen Bedienung dieses Bereiches, wenn die Anwendung auf Touch-Geräten ausgeführt werden soll.

Neben diesem Problemfall muss das bisherige Oberflächendesign für die einzelnen Komponenten für eine Fingerbedienung ausgelegt werden. Ebenfalls ist bei der Komponentenkonzeption auf den Wegfall des Mouse-Over-Effektes sowie eine stärkere visuelle Reaktion der Anwendung auf eintretende Berührungen zu achten. Diese und weitere Aspekte werden in einem Konzept für die Erstellung von Multitouch-fähigen Flex-Komponenten berücksichtigt.

4 Konzept

In diesem Kapitel wird ein Konzept entstehen, in dem Multitouch-fähige Flex-Komponenten hinsichtlich ihres Designs und Interaktionsmöglichkeiten definiert werden. Weiterhin wird ein Ansatz für die Umsetzung dieser Definitionen mit den Flex-internen Methoden und Eigenschaften gegeben. Der Bereich, auf den sich dieses Konzept konzentriert, ist das *Auswahl*-Fenster in der *Detailansicht* von Cyber. Für eine bessere Übersicht wird dieses Fenster in drei aufeinander aufbauende Unterkategorien geteilt. Abbildung 4.1 zeigt die drei verschachtelten Bereiche: Fenster, Tabelle und Zeitstrahl. Jeder dieser Bereiche ist damit ein gleichwertiger Teil des Konzeptes.

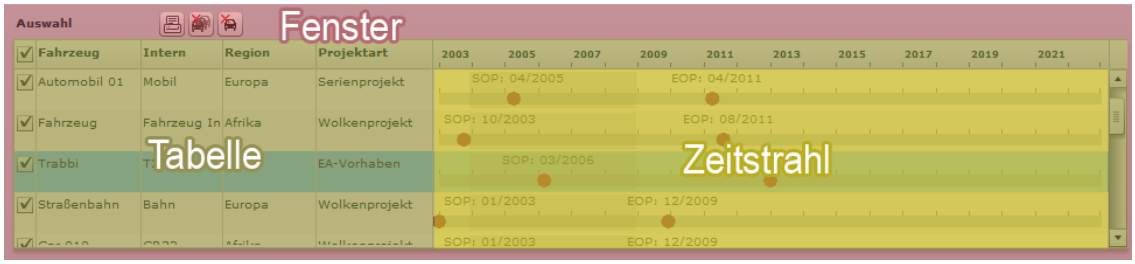

Abbildung 4.1: Das *Auswahl*-Fenster wird in drei verschachtelte Bereiche geteilt

Das Fenster *Auswahl* listet alle Fahrzeugprojekte eines Szenarios in einer Tabelle auf und dient weiterhin zum Auswählen eines oder mehrerer Projekte. Außerdem wird dem Benutzer ermöglicht, die auf einem Zeitstrahl abgebildeten SOP, EOP und GP in der entsprechenden Tabellenzeile zu verschieben. Bedient wurde das Fenster bislang mit der gewohnten Maussteuerung. Bei speziellen Funktionen, wie dem Hinzufügen einer neuen GP im Zeitstrahl musste die Tastatur zu Hilfe genommen werden. Mit Checkboxen, Schaltflächen und einer Tabelle weist das *Auswahl*-Fenster verschiedene Formen der Informationsdarstellung und Funktionenauswahl auf. Bestehend aus neuen und übernommenen Elementen werden auf Grundlage dieses Konzeptes Komponenten entstehen, die auf Multitouch ausgelegt und auf den dafür vorgesehenen Geräten bedienbar sind. Dabei spielen auch Gesten eine wichtige Rolle, die mit den Fingern dargestellt werden und vom System erkannt werden müssen.

4.1 Fenster

Das Fenster-Element des ausgewählten Bereiches besteht aus den Flex-Komponenten Panel und Button. Diese sollen sich auch weiterhin in der Cyber-Anwendung befinden, müssen aber für einen optimalen Gebrauch einigen Anpassungen unterzogen werden.

4.1.1 Interaktion

Schaltflächen

Für eine Nutzung mit den Fingern werden die drei bestehenden Buttons in ihrer Breite und Höhe vergrößert. Damit bereitet es dem Nutzer weniger Probleme, einen gewünschten Button auszuwählen. Für die optimale Größe einer Schaltfläche in einer Touch-fähigen Anwendung gibt es viele Richtlinien. Apple beispielsweise bietet die für iPhone-Apps relevanten Standard-Komponenten in einer Größe von 44 x 44 Pixeln an [6]. Bei einer Auflösung von 163 Pixel pro Zoll entsprechen 44 Pixel gerade einmal knapp 7mm. Dabei greifen Entwickler von mobilen Apps gern auf ein weit verbeitetes Prinzip zurück: nicht nur die Schaltfläche an sich ist berührbar sondern auch ein gewisser für den Nutzer nicht sichtbarer Bereich um sie herum ebenfalls. So wird die Fläche zum Auswählen eines Buttons nochmals vergrößert, was die Bedienung zunehmend erleichert. Allerdings kann diesem Ansatz nur nachgegangen werden, wenn es die Anzahl und Abstände der Bedienelemente zulassen. Forschungsgruppen der University of Oulu sowie der University of Maryland haben sich in einer Studie ebenfalls mit diesem Thema beschäftigt. Sie kamen zu dem Entschluss, dass eine durchschnittliche Größe von 9,2mm bis 9,6mm für berührbare Elmente, die mit einem Finger zu bedienen sind, ausreichend ist [1]. Ab dieser Größe nahmen sie keine wesentliche Reduzierung einer präzisen Auswahl der Komponenten mehr war.

Anders als die beiden vorangegangen Ansätze verfolgt Microsoft auf der Microsoft Developer Network (MSDN)-Seite eine Richtlinie, die die Schaltflächengröße von der gesamten Touch-Funktionalität der Anwendung abhängig macht [11]. So gibt es die Zustände *touchable* und *touch-enabled*, die ein Programm auszeichnen können. Mit *touchable* werden jene Anwendungen beschrieben, deren Bedienelemente aufgrund ihrer Größe einfach und präzise selektierbar sind. Diese sollten eine Mindestgröße von 24 mal 24 Pixeln aufweisen. Gesten sind dabei optional und Eingaberäte wie Tastatur sowie Maus können weiterhin verwendet werden. Als *touch-enabled* bezeichnete Anwendungen unterscheiden sich dadurch, dass die am meisten genutzten Bedienelemente mindestens 40 Pixel breit und hoch sind. Außerdem werden alle bekannten Gesten voll unterstützt. Auf der Grundlage dieser Ansätze wird in Cyber jenen Komponenten, die mit dem Finger berührt werden können, eine Größe von mindestens 30 mal 30 Pixel zugewiesen. Die 30 Pixel entsprechen bei einer Bildschirmauflösung von 1280 mal 800 Pixel etwas mehr als 8mm. Die Angabe einer Größe im Pixelmaß ist allerdings immer relativ. Bei einer höheren Auflösung erscheinen die Elemente kleiner, auf Bildschirmen mit kleinerer Auflösung größer. Doch auch auf Bildschirmen mit einer Breite von 1680 Pixeln sind 30 Pixel-breite Bedienelemente noch präzise auswählbar.

Das Problem, dass ein Finger beim Auswählen eines Buttons dessen Oberfläche teilweise verdeckt, bleibt zwar bestehen, wird aber durch das Aufleuchten einer berührten Schaltfläche verringert. Damit bleibt die Bindung zwischen Benutzer und dem berührten

Element erhalten. Außerdem können auch weiterhin die Data Tips angezeigt werden, die sonst erscheinen, wenn der Benutzer den Cursor über einen der Buttons bewegt. Für den Wegfall des Mouse-Over-Effektes bietet Flex dazu zumindest eine Alternative. Legt der Benutzer ähnlich wie in Abbildung 4.2 seinen Finger auf die Oberfläche und bewegt ihn ohne abzusetzen über den Button, so leuchten die Data Tips auf. Dieses Ereignis wird in dem Multitouch-Application Programming Interface (API) als Touch-Roll-Over beschrieben und kann so in einigen Fällen den ehemaligen Mouse-Over ersetzen.

Abbildung 4.2: Mit einem Touch-Roll-Over können auch weiterhin Data Tips angezeigt werden (vgl. Quelle [18])

Anfasser

Bei der Panel-Komponente kann ein im Bereich von Multitouch-Anwendungen aber auch in aktuellen Microsoft Office-Produkten bekannter Ansatz verfolgt werden. Sogenannte Adorner oder auch Anfasser dienen einem Benutzer, das gewählte grafische Element in dessen Größe und Rotation zu ändern. In Cyber muss der Benutzer zur Anzeige der Anfasser mit seinem Finger einen der Divider in den Zwischenräumen der *Detailansicht* berühren. Dabei ermöglichen nur die beiden an das *Auswahl*-Fenster angrenzenden Divider einen Aufruf der Funktion. Dadurch leuchten rund um das *Auswahl*-Fenster an der Panel-Komponente drei Anfass-Punkte auf. Wird einer dieser Punkte bewegt, ändert sich die Größe des Panels anhand des Anfasser. Das Panel reagiert damit direkt auf die Berührungen des Benutzers. Beim Wählen einer anderen Stelle der Applikation blenden alle Anfass-Punkte aus. Eine im Bereich der Touch-fähigen Anwendungen weit verbreitete Funktionalität kann auch hier einen großen Nutzen erbringen. Somit ist der Nutzer stets in der Lage, das Erscheinungsbild des *Auswahl*-Fensters hinsichtlich dessen Größe effizient nach seinen Wünschen anzupassen.

Divider

Ähnlich wie die Schaltflächen müssen sich auch die Divider-Komponenten vergrößern. Sowohl der Skin des eigentlich sichtbaren Divider-Anfassers als auch die Abstände zwischen den Panels werden dafür geändert. Dabei orientiert sich das Aussehen nicht mehr an den üblichen waagerechten Linien sondern an drei nebeneinanderliegenden Kreisen, die in Verbindung zu den aufrufbaren Anfassern stehen. Der Benutzer stellt mit

dem neuen Divider-Design eine Verbindung zu den kreisförmigen Adornen her, die er über diese Komponente aufrufen kann.

4.1.2 Software

Schaltflächen

Bei der Umsetzung der Interaktionsziele im programmiertechnischen Sinn muss lediglich die Panel-Komponente näher betrachtet werden, da die Größe der Schaltflächen über die *width*- und *height*-Eigenschaften bestimmbar sind. Die Data Tips der Buttons erscheinen ohne das Abfragen von speziellen Touch-Roll-Over-Events. Dazu werden Berührungen dieser Komponente weiterhin als standardmäßige Mouse-Events behandelt.

Anfasser

Um den Ansatz der Adorner-Funktionalität zu implementieren, wird mit der Erweiterung ObjectHandles gearbeitet. Diese Klassensammlung wird von dem Autor des Projektes »Rogue-Development.com« Marc Hughes kostenfrei auf GoogleCode zur Verfügung gestellt und steht unter der MIT-Lizenz[19]. Damit ist ein kostenloser Einsatz von ObjectHandles möglich. Mit Hilfe von ObjectHandles in der Version 2.0 können Container grafische Objekte aufnehmen, die sich durch eine definierte Anzahl von Anfassern in ihrer Größe anpassen lassen. Nach Implementierung in das aktuelle Flex-Projekt können sogenannte ObjectHandles-Objekte neu instanziiert werden, welche anschließend als Container wirkend grafische Elemente aufnehmen. Durch die Zuweisung derselben Größe für Container und grafischem Element wirken sich alle Änderungen am ObjectHandles-Objekt identisch auf das Kind-Element aus. Im Beispiel der Panel-Komponente bedeutet dies, dass sich das Panel an den Änderungen des ObjectHandles-Containers ausrichtet. Event-Listener reagieren auf eintretende Berührungen des Box-Dividers in den Zwischenräumen zweier benachbarter Fenster. Wird ein Divider mit dem Finger berührt, erscheinen entlang des *Auswahl*-Fensters die Anfasser. Nach definierten Angaben wird es anschließend möglich sein, diese Anfasser zu ziehen. Die Event-Handler werten diese Änderungen aus und begrenzen beispielsweise die maximale Breite auf eine vorgegebene Größe.

Die Abfrage des berührten Dividers ist in Flex etwas aufwändig. Da nur die Divider die Anfasser erscheinen lassen sollen, die rund um das Fenster *Auswahl* angeordnet sind, darf der unterste Divider diese Funktionalität nicht unterstützen. Deshalb wird bei der Berührung einer *VDividedBox*-Komponente die Anzahl der aufgeführten Divider

[19] Die MIT-Lizenz stammt aus dem Massachusetts Institute of Technology und erlaubt die Wiederverwendung von Software.

errechnet. Über eine Schleife und der Methode *getDividerAt(i)* kann über die *name*-Eigenschaft des Zielobjektes ermittelt werden, um welchen der beiden Divider es sich bei dem berührten Objekt handelt.

Divider

Da sich das Aussehen der Divider-Komponenten an den gewählten Größen für Bedienelemente von 30 Pixeln orientieren soll, wird ihr Skin individuell gestaltet. Über die *Stylename*-Eigenschaft kann der entspechende Stil mittels *divider-skin* oder *divider-alpha* eingestellt werden. Die Grafik für den eigentlich Skin[20] liegt dabei in einer eingebetteten Bilddatei vor und dient dem Benutzer später zur Verschiebung des Dividers.

4.1.3 Screendesign

Schaltflächen

Neben dem Interaktionsdesign muss auch das Interfacedesign bei der Konzipierung einer Software beachtet werden. Mit dem Interfacedesign wird hauptsächtlich die Gestaltung von Benutzeroberflächen beschrieben. „Das Interfacedesign definiert, steuert und ermöglicht den Dialog und die Dialogfähigkeit zwischen Mensch und Maschine [...]“ [29], erläutert Torsten Stapelkamp bei der Entwicklung von Interaction- und Interfacedesign. Eine gute Dialogfähigkeit wird im Fall von Touch-fähigen Anwendungen auch durch Oberflächen, die für den Benuter anfassbar wirken, ermöglicht. Neben einer plastischen Darstellung muss dabei eine angepasste Objektgröße vorliegen. Die Buttons der Fenster-Komponente müssen für einen optimalen Gebrauch nur in ihrer Größe geändert werden. Die Abbildung 4.3 verdeutlicht diesen Unterschied.

Abbildung 4.3: Links - die bisherigen Buttons, rechts - die Buttons nach ihrer Vergrößerung (andere Elemente sind für diese Abbildung ausgegraut)

Weiterhin bleiben die Flex-internen Data Tips implementiert. Aus Abbildung 4.4 wird ersichtlich, dass sie immer dann erscheinen, wenn ein bereits aufgelegter Finger die Schaltflächen berührt.

[20] Mit einem Skin kann innerhalb von Flex die Oberfläche einer Komponente neu definiert werden. Skins liegen oftmals als eingebettete Grafikdateien vor.

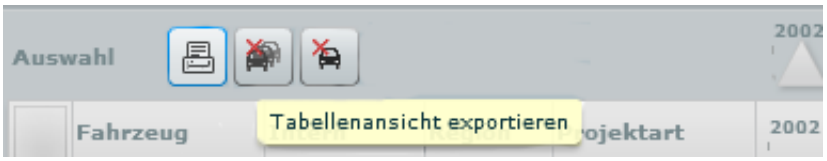

Abbildung 4.4: Die Data Tips erläutern kurz die Funktion der Schaltfläche

Anfasser

In den folgenden Abbildungen wird die Anordnung und Erscheinung der Anfasser definiert. Wie in Abbildung 4.5 erkennbar, ordnen sich drei kreisförmige Anfasser links und rechts unten sowie rechts oben an. Diese Festlegung erfolgt, da die Größer der Panel-Komponente nur nach rechts und nach unten veränderbar sein darf. Verschiebungen nach links werden nicht unterstützt, da sich das Panel bereits an der linken Seite der gesamten Anwendung orientiert. Ebenso ist der Abstand zum oberen Ende der Anwendung auf die ButtonControlBar-Komponente der Filterfunktionen festgelegt.

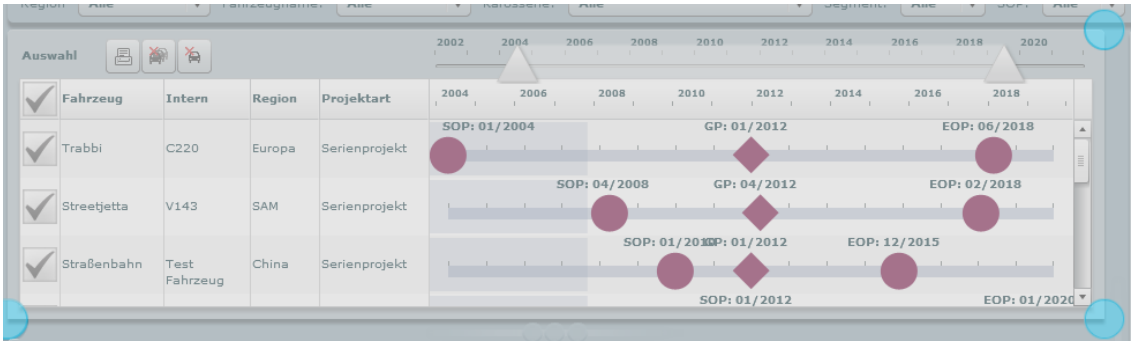

Abbildung 4.5: Die Adorner erscheinen an drei Eckpunkten des Panels

Die Kreise der Anfasser erscheinen zunächst in einem hellblauen Ton. Legt der Benutzer einen Finger auf eines dieser Objekte, ändert sich dessen Farbe, wie in Abbildung 4.6 erkennbar, hellgrün.

Abbildung 4.6: Links - der Anfasser im Normalzustand, rechts - Farbänderung bei Berührung

Divider

Damit die Divider auf berührungsempfindlichen Geräten bedient werden können, müssen sie sich in ihrer Objektgröße ändern. Durch diese Anpassung an die Fingergröße des Menschen sollte es dem Benutzer leichter fallen, diese Komponenten auszuwählen. Abbildung 4.7 zeigt, dass sich ihre Form nun aus drei anfassbaren Kreisen aufbaut.

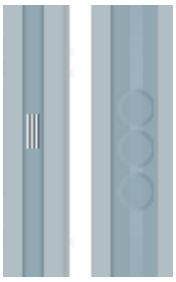

Abbildung 4.7: Links - der bisherige Divider, rechts - der individuelle Divider

4.2 Tabelle

In der Tabelle finden sich einige Komponenten wieder, die für eine Anwendung in einer Multitouch-Applikation interessant erscheinen. Dabei sticht vor allem die Flex-DataGrid-Komponente heraus. In ihr werden alle aktuellen Fahrzeugprojekte aufgelistet, die zur weiteren Darstellung ausgewählt werden. Weiterhin umfasst die Komponente *Tabelle* die in der ersten Spalte aufgeführten Checkboxen. Mit der Einführung neuer Bedienkonzepte wird neben der Anpassung grafischer Elemente auch eine Weiterentwicklung oder sogar Neuentwicklung der Interaktionsdesigns angestrebt. Die Interaktionsmöglichkeiten in der DataGrid-Komponente beschränkten sich bisher auf das Auswählen eines oder mehrerer Projekte unter Benutzung der Maus. Zur Multiselektion musste zusätzlich die *<STRG>*-Taste zur Hilfe genommen werden.

4.2.1 Interaktion

Scrollen anhand der Touch-Events

Um künftig das Scrollen in der DataGrid bei einer hohen Anzahl an Einträgen zu vereinfachen, wird dafür eine bestimmte Geste Verwendung finden. Zwei parallel aufgelegte und nach unten oder oben verschobene Finger werden dazu als Scroll-Geste definiert. Hierbei wurde sich auf zwei Finger festgelegt, um diese Geste strikt von einer Ein-Finger-Auswahl oder einer Drag-And-Drop-Bewegung zu trennen. Mit Hilfe der Scroll-Geste soll es dem Benutzer erleichtert werden, sich wie in Abbildung 4.8 durch die aufgelisteten Projekte zu bewegen. Die Verwendung des bekannten Scrollbalkens kann auf Touch-Oberflächen zu Problemen führen. Denn die Auswahl und Positionierung des Balkens ist stark von dessen Objektgröße abhängig. Dies hätte mit einer Vergrößerung des Scrollbalkens auf Kosten des vorhandenen Platzes der Oberfläche eingedämmt werden können. Um beim Scrollen in der Tabelle keine neue Zeile auszuwählen, wird die aktuelle Zeilenauswahl beibehalten. Dafür erfolgt eine Deaktivierung der Tabelle während des Scroll-Vorganges. Nach dem Absetzen der Finger ist auch die Tabelle wieder aktiv, ihre Scroll-Position hat sich aber geändert. Der Nachteil an der Variante der deaktivierten Tabelle ist, dass der Scroll-Balken verschwindet. Damit wird dem Benutzer ein Bezugspunkt zur aktuellen Scroll-Position genommen. Auch die Ein-

ordnung, wann das Tabellenende erreicht ist, erweist sich als schwieriger. Daher muss sich der Benutzer an einem neuen Bezugspunkt in Form einer der Tabellenzeilen orientieren.

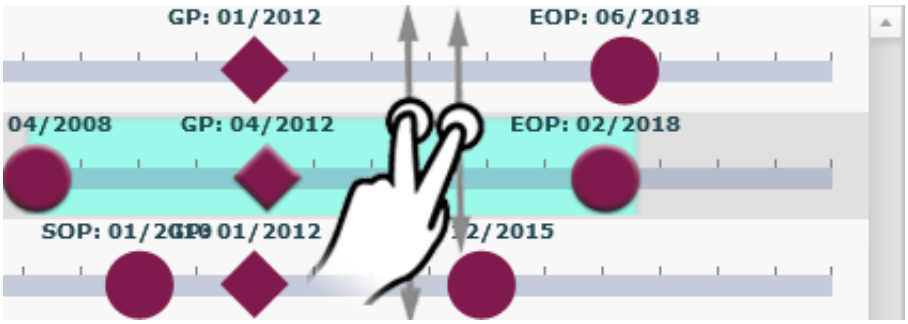

Abbildung 4.8: Mit der Scroll-Geste kann der Benutzer die Scroll-Position der Tabelle ändern (vgl. Quelle [18])

Zoomen mit spezieller Komponente

Neben der Scroll-Geste wird dem Nutzer zur Interaktion das Zoomen in den Zeitstrahl ermöglicht. Diesbezüglich gab es Überlegungen, ob die Implementierung einer Zoom-Geste in der Tabelle den größten Nutzen für die Anwendung bringen würde. Der Benutzer hätte dabei mit zwei Fingern scrollen und zoomen können. Doch dies hätte im Gegenzug auch die Komplexität der Anwendung für den unerfahrenen Benutzer deutlich erhöht. Eine klare Trennung zwischen den beiden Funktionen ist daher nötig. Außerdem sind Gesten, so intuitiv sie für den Entwickler auch erscheinen, für gewisse Nutzergruppen gänzlich unbekannt. In einer Studie der User Interface Design GmbH (UID) namens *Weltweit berührt* [3] wurde festgestellt, dass für zahlreiche Funktionen von Testpersonen oft unterschiedliche Gesten bevorzugt wurden. Ein Beispiel war auch dort das Zoomen in ein Objekt. Nur weniger als die Hälfte aller Testpersonen hat dazu zwei Finger über dem Objekt gespreizt. Andere tippten das Objekt einmal oder mehrmals an. Allerdings verwendeten Personen, die bereits Erfahrungen mit Multitouch-Oberflächen aufweisen konnten, oftmals die bekannte Pinch-Geste. Auffällige regionale Unterschiede konnten dabei nicht hervorstechen. Dieses Szenario ist auch auf andere Gesten und Bewegungen übertragbar. So kann man schlussfolgern, dass unterstützte Gesten, so primitiv sie auch erscheinen, von einem Erstbenutzer ohne Vorkenntnisse nicht immer ohne Weiteres korrekt angewandt werden. Liegen allerdings bereits Erfahrungen mit dieser oder ähnlichen Multitouch-Anwendungen vor, so ist die Wahrscheinlichkeit hoch, dass bestimmte Gesten bereits erlernt wurden und man diese beim Nutzer voraussetzen kann.

Um unter Anderem die Zahl der Gesten zu reduzieren, fiel die Entscheidung auf die Implementierung von berührungsempfindlichen Komponenten. Mit ihnen kann der Benutzer das angezeigte Projektintervall der Fahrzeugprojekte beeinflussen. Anhand zweier pfeilförmiger Objekte sind die obere und untere Intervallbeschränkungen veränder-

bar. Ähnlich wie bei einem Slider liegen diese über einem Hintergrund und lassen sich nur waagerecht verschieben. Legt man einen Finger auf einen der Pfeile und zieht ihn entlang des Hintergrundes, bewegt sich der Pfeil anhand der Position der Berührungspunktes. Unterstützt wird hierbei die Funktionalität, dass beide Pfeile gleichzeitig und unabhängig voneinander verschoben werden können. Dies wird auch als Multi-Drag-And-Drop bezeichnet. Die Abbildung 4.9 verdeutlicht das Prinzip. Werden die Pfeile losgelassen und keines der beiden Objekte mehr berührt, wird das neue anzuzeigende Intervall berechnet. Mit größeren Abständen zwischen den einzelnen Jahren steigt die räumliche Distanz von zwei angezeigten Produktionsabschnitten, beispielsweise zwischen zwei GPs. Neben einer daraus folgenden effizienteren zeitlichen Einschätzung der Produktionsabschnitte überlagern sich die angezeigten Informationen über den jeweiligen Punkten nicht mehr. Um zusätzlich den Bezug zwischen den Pfeilkomponenten und dem dargestellten Intervall zu vergrößern, verbindet ein heller Bereich die zwei bewglichen Objekte miteinander. Dieser Bereich steht für den in der Tabelle angezeigten Zeitabschnitt. Die Möglichkeit, in den Zeitstrahl zu zoomen, gibt dem Benutzer eine bessere Übersicht über die zeitliche Einordnung von Produktionsmerkmalen und verstärkt das Gefühl, das Erscheinungsbild der angezeigten Daten nach Belieben anzupassen.

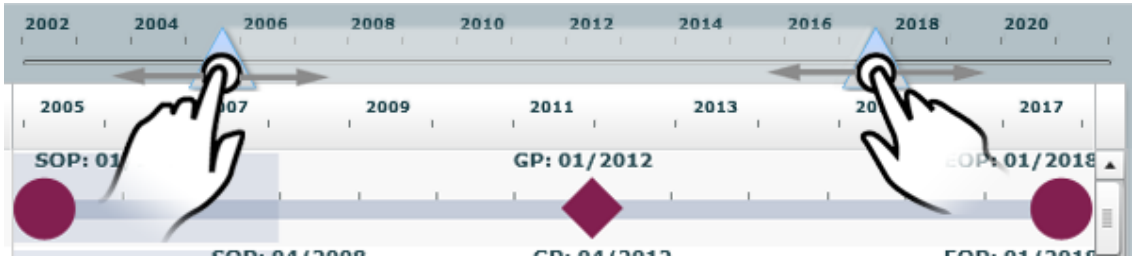

Abbildung 4.9: Die zwei Pfeilkomponenten sind gleichzeitig verschiebbar (vgl. Quelle [18])

Multiselektion in der DataGrid

Überarbeitet wird auch das Auswählen einer Zeile in der DataGrid. Bislang klickte der Nutzer einmal in die Zeile und erwirkte damit seine Auswahl. Bei der Verwendung von berührungsempfindlichen Geräten kann diesem Ansatz nicht immer nachgegangen werden. Da Berührungen unter Umständen als Mausklick interpretiert werden, würde dies am Beispiel der DataGrid mit folgenden Szenario gleichkommen. Ein Benutzer legt seinen Finger auf die Tabelle und bewegt ihn von oben nach unten. Als Reaktion könnte jede Berührung einer Zeile das Auswählen des aktuellen Listenelementes zur Folge haben, was der Benutzer nicht beabsichtigt hat. Daher erfolgt das Auswählen künftig mit der bekannten Tap-Geste. Bei einem Tap tippt man die Zeile kurz an und wählt sie damit aus. Ein Aufsetzen und Ziehen des Fingers wird ignoriert. Es wirkt sich daher nur aufeinanderfolgendes Auf- und Absetzen des Fingers aus. Dieses Verhalten wird mit dem Flex 4-SDK automatisch in einer DataGrid-Komponente angewandt. Um weiterhin die gleichzeitige Auswahl mehrerer Zeilen in der DataGrid zu realisieren, bleibt eine bereits selektierte Zeile beim Antippen einer weiteren aktiv. Im Umkehrschluss lässt sich

ein bereits ausgewähltes Element mit einem erneuten Tap darauf wieder deaktivieren. Mit diesem Verhalten erleichtert man dem Benutzer die Handhabung von Auflistungen. Intuitiv lassen sich Projekte an- und abwählen, was eine weniger komplexe Bedienung der Tabelle garantiert.

Checkbox als Item-Renderer

Mit den Checkboxen befinden sich typische, aus Web-Anwendungen bekannte Interaktionselemente in der Tabelle wieder. Diese werden weiterhin verwendet, müssen aber in ihrer Größe und Erscheinung geändert werden. Denn zu kleine Checkboxen machen es dem Benutzer unmöglich, sie mit dem Finger auszuwählen. Auf diese Komponenten zu verzichten ist in der Anwendung nicht angebracht. Denn mit ihrer Hilfe wird es dem Benutzer erleichert, die Projekte anzuzeigen, die in den *Summe*- und *Selektion*-Fenstern dargestellt werden. Da bei der Vergrößerung der Standard-Checkboxen in Flex unscharfe Oberflächen entstehen, werden neben einer geänderten Größe auch ein neuer Skin die Erscheinung der Checkboxen in Cyber prägen.

Texteingabe mit virtueller Tastatur

Eine neue Interaktionsmöglichkeit stellt die Benutzung einer virtuellen Tastatur dar. Mit ihrer Hilfe ist es möglich, Daten in der Cyber-Anwendung ohne Anschluss einer physikalischen Tastatur einzugeben. Vorerst wird die virtuelle Tastatur des Betriebssystems verwendet. In zukünftigen Entwicklungen soll in Cyber eine eigene Tastatur implementiert werden, da auch Systeme ohne aktivierte virtuelle Tastatur berücksichtigt werden müssen. Am Beispiel von Windows 7 wird die Tastatur in ihrem Ruhezustand am linken Rand angedeutet und kann mit einer Berührung vollständig eingeblendet werden.

Um Tastatureingaben im ausgewählten Bereich von Cyber zu ermöglichen, wird der Benutzer künftig die Tabellen-Zellen, die reinen Text enthalten, direkt ändern können. Dabei muss auf eine Zelle einer bereits aktiven Zeile wie in Abbildung 4.10 doppelt getippt werden. Anschließend wird das Textfeld umrahmt und es erscheint ein Cursor hinter dem letzten Zeichen. Per Berührung des daraufhin aufblendenden Tastatursymbols erscheint die interne virtuelle Tastatur des Betriebssystems über der Anwendung und ist zur Eingabe bereit. Nach Abschluss der Eingabe durch Betätigen der virtuellen *<ENTER>*-Taste endet der Bearbeitungsvorgang und die Zelle liegt mit der vorgenommenen Änderung vor. Damit ist der Benutzer zum Ändern dieser Datensätze nicht mehr auf die Verwendung des Menüs zum Editieren eines Projektes angewiesen. Daraus resultiert neben einer Zeitersparnis auch eine bessere Übersicht. Des Weiterhin gilt das direkte Bearbeiten von Textfeldern als sehr bekannt und ist als intuitiver zu bewerten, als die Verwendung spezieller Formulare. Der vorgesehene Einsatzbereich der Tastatur geht über die Tabelle hinaus. In späteren Entwicklungsstufen soll sie auch in anderen Fenstern und Formularen als primäres Eingabemedium dienen.

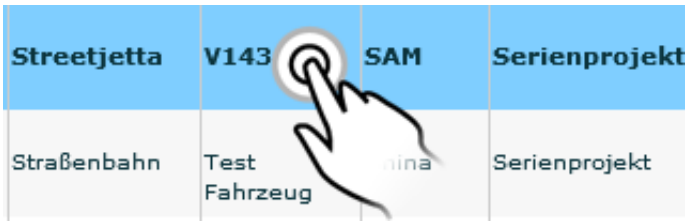

Abbildung 4.10: Mit einem Double-Tap auf ein Textfeld wird dieses bearbeitbar (vgl. Quelle [18])

4.2.2 Software

Scrollen anhand der Touch-Events

Um die Funktion des Scrollens zu realisieren, müssen die Daten der von dem Flex Framework bereitgestellten und erkannten Ereignisse der Touch-Event-Klasse verarbeitet werden. Auf eine bestehende Geste aus dem Framework kann hierbei nicht zurückgegriffen werden, da Scroll-Gesten von der API nicht bereitgestellt werden. Bei dem Einsatz der Touch-Events bietet Flex die Differenzierung eines Ereignisses in die Phasen *Begin*, *Move* und *End* an, die äquivalent für den Beginn der Berührung, die Bewegung und das Absetzen eines Fingers eingesetzt werden. Für jede dieser Phasen kann mit Hilfe von Event-Listenern auf das Ereignis reagiert werden. Legt man einen Finger auf die Tabelle, verarbeitet eine darauf angesetzte Methode die Informationen zur Position dieses Punktes. Erst wenn ein weiterer Finger die Oberfläche berührt und beide bewegt werden, wird anfangs ein Mittelwert aus den beiden vertikalen Werten der ursprünglichen Berührungspunkte errechnet. Dadurch wird es dem Nutzer erlaubt, seine Finger sowohl parallel zueinander als auch leicht versetzt zu verschieben. Durch die Bewegung folgt ein Aufruf der TouchMove-Event-Handler. Neben der Positionsspeicherung vergleichen diese die alte Position mit der aktuellen und verändern die Scrollposition der DataGrid mit Hilfe der Eigenschaft *VerticalScrollPosition* dementsprechend. Weiterhin werden die maximalen und minimalen Scroll-Positionen mit Null sowie der Eigenschaft *MaxVerticalScrollPosition* begrenzt.

Mit dem baldigen Erscheinen von Flash Builder Burrito in Verbindung mit Adobe Flex Hero bietet Flex die Möglichkeit, standardmäßig scrollbare Elemente zu verwenden. Dabei kann durch List-Komponenten ohne weiteren Programmieraufwand mit einem Finger gescrollt werden. Scroll-Balken erscheinen dabei nur noch direkt beim Scrollen. Dasselbe Verhalten kann auch mit sogenannten Scroller-Komponenten erzielt werden. Ein Scroller wirkt dabei wie ein Container, der eine Anzahl an grafischen Elementen aufnimmt. Übersteigen die aufgenommenen Elemente die Bühnengröße, erscheinen die Komponenten außerhalb der Bühne erst, wenn man durch die Scroller-Komponente scrollt. Da sich diese Elemente bislang eher für mobile Projekte eignen, werden sie für diese Bachelorarbeit vernachlässigt. Außerdem ist die Geste zum Scrollen auf nur einen Finger begrenzt. Des Weiteren liegt zum Zeitpunkt der Bachelorarbeit nur eine Beta-Version des Flash Builders Burrito vor, die noch nicht frei von Fehlern ist.

Zoomen mit Pfeil-Komponente und Multidrag

Um in die Auflistung der Objekte zu zoomen und damit das anzuzeigende Intervall zu ändern, müssen die dafür vorgesehenen Pfeil-Objekte verschoben werden. Bei diesen Objekten handelt es sich um dynamisch geladene Bilder, die zusammen in einer Action Script-Klasse von je einem Canvas aufgenommen wurden. Sobald ein Finger auf den Canvas-Komponenten aufliegt, fängt ein Event-Listener das TouchBegin-Event ab und startet eine *StartTouchDrag*-Funktion. Wichtig für diese Funktion ist neben der Zuweisung des zu bewegenden Objektes auch die Übergabe der *TouchPointID* des betreffenden Touch-Events. Ohne diese Informationen kann es vorkommen, dass beim Auf- und Ablegen eines Fingers nicht festgestellt wird, welches Objekt bei mehreren Berührungen verschoben werden soll. Durch diese Funktionalität ist der Benutzer in der Lage zwei sichtbare Elemente auf der Oberfläche gleichzeitig und unabhängig voneinander zu verschieben.

Wenn die Berührung der Pfeile unterbrochen wird, berechnet eine Funktion aus den horizontalen Koordinaten der beiden Objekte das neue Intervall für die Slider des Zeitstrahles. Unter Berücksichtigung der aktuellen Breite des *Auswahl*-Fensters werden sowohl die kleinste anzuzeigende Jahreszahl sowie die Gesamtanzahl aller anzuzeigenden Jahre kalkuliert. Außerdem beendet die Funktion *StopTouchDrag* die aktuellen Drag-And-Drop-Vorgänge. Diese Funktion muss anhand eines TouchEnd-Event-Listeners aufgerufen werden, der auf alle beendeten Berührungen der gesamten Applikation eingeht. Denn der Benutzer soll die Pfeile auch außerhalb ihrer Objektbegrenzungen loslassen können.

Als alternative Interaktionsmöglichkeit hätte ebenfalls eine Zoom-Geste in diese Anwendung implementiert werden können. Aufgrund der im Kapitel 4.2.1 aufgeführten Gründe wurde unter Anderem aufgrund der steigenden Komplexität darauf verzichtet. Ein weiteres wesentliches Problem bestand darin, dass bei unerfahrenen Benutzern Gesten wie das Zoomen nicht automatisch vorausgesetzt werden können. Flex bietet zwar insgesamt sechs vordefinierte Gesten, allerdings ist deren Einsatz mit Schwierigkeiten verbunden. Eine dieser Problematiken betrifft die Abhängigkeit der unterstützten Gesten von dem Betriebssystem. Dadurch ist die Geste des Zwei-Finger-Taps zwar bei einer Ausführung auf einem PC mit Windows 7 erkennbar, auf Geräten mit Mac OS wird diese allerdings ignoriert. Die einzigen Gesten, die sowohl auf Geräten mit Windows 7 als auch mit Mac OS gleichermaßen erkannt werden, sind Pan, Rotate und Zoom. Der zweite wesentliche Kritikpunkt betrifft das gleichzeitige Auftreten von mehreren Gesten. Wird beispielsweise in ein Bild gezoomt und in der Bewegung die Finger ähnlich der Rotieren-Geste gedreht, dann wird nur das Zoomen verarbeitet. Dementsprechend kann immer nur eine Flex-interne Geste allein auftreten.

Bei einem alternativen Lösungsweg würde man versuchen nur die Zoom-Geste für die Anwendung zu implementieren und die restlichen Interaktionsmöglichkeiten mit den

Touch-Events zu realisieren. Doch auch dabei stößt der Entwickler auf ein wesentliches Problem. In einer Anwendung können entweder Touch-Events oder Gesture-Events abgefragt werden. Der Einsatz beider Event-Arten wird nicht unterstützt. Auch deswegen muss möglichst vor dem Festlegen der Event-Listener der Eingabemodus mittels *Multitouch.InputMode* eingestellt werden. Dafür bietet Flex die Wahl zwischen *None*, *Gesture* und *Touch_Point*. Der erste der drei Modi wird für Geräte verwendet, die keinerlei Touch-Unterstützung aufweisen. In diesem Modus werden nur Maus-Events abgefragt. Mit *Gesture* kann der Entwickler die vorgegebenen Gesten verwenden, ist aber auf diese begrenzt. Anhand des dritten Modus *Touch_Point* können alle auftretenden Berührungen abgefragt und verarbeitet werden. Diese Einstellung wurde auch für Cyber festgelegt. Die Informationen dieser Touch-Events erhält die Flex-Anwendung von dem Betriebssystem. Deshalb können mit dem Flex-Framework nur die Touch- und Gesture-Events verwendet werden, die von dem zugehörigen Betriebssystem unterstützt werden. Im Modus *Touch_Point* sind gewünschte Gesten allerdings eigenhändig zu programmieren.

Da die Programmierung eigener Gesten mit großem Aufwand verbunden sein kann, bietet sich ein Blick in Projekte wie »Touchlib« an. Dort findet man Klassen, wie die *RotatableScalable.as*. Mit ihr ist es möglich, Objekte die von der Klasse *DisplayObject* erben, gleichzeitig zu drehen und zu zoomen. Da »Touchlib« nicht mit den Flash-internen Touch-Events arbeitet, müssen deren Klassen auf diese Ereignisse abgestimmt werden. Dafür bietet Cynergy Systems-Softwareentwickler Andrew Trice auf »cynergy blogs« einige Einblicke, wie beispielsweise die *RotatableScalable*-Klasse in dem Flex-Framework verwendet werden kann [30]. Ebenfalls zu empfehlen ist das Multitouch-Framework »GestureWorks« von Ideum. Es zeichnet sich vor allem durch seine rund 200 verschiedenen Gesten aus, darunter auch Fünf-Finger-Verschiebungen und Gesten zum Scrollen [16]. Allerdings traten beim Testen mit der Version 1.5.3 in der Flex-DataGrid-Komponente Probleme beim Auswählen von Zeilen auf. Inwieweit sich die im Dezember 2010 erschienene Version 2 von »GestureWorks« für Flex-Projekte eignet, konnte in dieser Bachelorarbeit noch nicht nachgewiesen werden. Außerdem ist der Einsatz dieses Frameworks nur dann sinnvoll, wenn in der zu entwickelnden Anwendung zahlreiche Gesten eingesetzt werden sollen, deren Entwicklung sich als kompliziert erweist. Da in der Cyber-Applikation aber keine große Anzahl an Gesten auftritt, konnte auf den Einsatz von »GestureWorks« verzichtet werden.

Multiselektion in der DataGrid

Das Auswählen einer Zeile erfolgt bei Auftreten eines TouchTap der Touch-Event-Klasse. Damit Zeilen beliebig an- und abwählbar sind, werden bestimmte Eigenschaften der DataGrid-Klasse geändert. Mit der *ctrlKey*-Eigenschaft kann einer DataGrid mitgeteilt werden, ob die Steuerung-Taste gedrückt wurde. Anhand dieses Wissens ist es möglich, der Applikation mitzuteilen, dass diese Taste gerade aktiv ist, obwohl dies nicht stattfindet. Dafür wurde die geschützte Funktion *selectItem* der DataGrid mit einem neu-

en Wert für *ctrlKey* überschrieben. So ist bei einem TouchTap künftig eingestellt, dass die Steuerung-Taste in einem aktiven Zustand ist und damit intuitiv Zeilen der Auswahl hinzugefügt und abgezogen werden können.

Soll hingegen eine einzelne Zeile ausgewählt werden, damit Änderungen bezüglich der Textfelder oder des Slider vorgenommen werden, muss der Finger für mehrere Sekunden auf einer Zeile liegen. Dies teilt der Applikation mit, dass die *ctrlKey*-Eigenschaft auf *false* gesetzt wurde, was wieder einem normalen Mausklick entspricht. Aus diesem Verhalten resultiert, dass in diesem Fall nur die aktuell berührte Zeile aktiv ist. Die vorher selektierten Zeilen befinden sich nun in einem inaktiven Zustand.

Checkbox als Item-Renderer

Die auf Berührung reagierenden Checkboxen können durch Anpassung des betreffenden Item-Renderers wesentlich geändert werden. Ein Item-Renderer ist in diesem speziellen Fall eine MXML-Komponente, die eine weitere Komponente wie eine Checkbox sowie Action Script-Code beinhaltet. Für jede Zeile in der entsprechenden Spalte der DataGrid wird dabei dieser Item-Renderer neu instanziiert, sodass er für alle angezeigten Zeilen gilt. Sollte beim Scrollen durch eine DataGrid eine vorher nicht sichtbare Zeile angezeigt werden, wird für diese keine neue Instanz des Item-Renderers erstellt, sondern der Item-Renderer der vorher dargestellten Zeile angewendet und damit recycelt. Durch Überschreibung der *set data*-Funktion werden jeder dieser Instanz die zugehörigen Informationen aus dem Datenprovider der DataGrid übergeben. Die Änderung des Checkbox-Stiles erfolgt durch Setzen der sechs verschiedenen Icons für die aktiven und inaktiven Checkbox-Zustände mit Hilfe der *StyleManager*-Eigenschaft.

Texteingabe mit virtueller Tastatur

Um den darzustellenden Text in der DataGrid in angepassten Textfeldern anzuzeigen, ist auch hierbei der Einsatz eines Item-Renderers notwenig. Im normalen Zustand wird der Text in einer Text-Komponente angezeigt, welche die Schriftgröße bei einem Antippen einer Zeile für alle Textfelder dieser Zeile ändert. Damit wird auf die Problematik eingegangen, dass der Nutzer durch Berührungen eine visuelle Reaktion der berührbaren Komponenten erhält. Der DoubleTap auf ein Textfeld wird mit Aktivieren der *doubleClickEnabled*-Variable ermöglicht. Ein *Click* wird dabei durch einen Tap ausgelöst. Nach einem DoubleTap erscheint an der Stelle der Text-Komponente eine TextArea mit dem identischen Inhalt. Der Vorteil dieser Variante ist, dass die TextArea bearbeitet werden kann. Die Text-Komponente eignet sich aber für den Gebrauch des normalen Anzeigen eines Textes durch die Eigenschaft, dass sich der Hintergrund ihrer gesamte Zelle im aktiven Zustand ändert. Nach Eingabe des Textes kann mit dem Antippen einer beliebigen Stelle außerhalb der TextArea oder durch Bestätigung mit der *<ENTER>*-Taste der virtuellen Tastatur die Änderung vorgenommen werden. Anschließend liegt wieder die

Text-Komponente mit dem überarbeiteten Inhalt vor.

Um zu gewährleisten, dass bei Vorhandensein einer virtuellen oder physischen Tastatur diese auch genutzt wird, garantieren Abfragen zu den Eigenschaften *hasVirtualKeyboard* sowie *physicalKeyboardType* einen korrekten Einsatz. Erst wenn beide Eigenschaften zu erkennen geben, dass das System nicht auf eine eigene Tastatur zurückgreifen kann, muss eine virtuelle Tastatur zum Einsatz kommen, die nur für diese Anwendung implementiert wurde.

4.2.3 Screendesign

Pfeilkomponente

Der Pforzheimer Professor für Wirtschaftsinformatik Stephan Thesmann beschreibt das Interfacedesign in seinem Buch über das Design multimedialer Webanwendungen als ein zentrales Element, „[. . .]das den Benutzer, die zu lösende Aufgabe und das dafür benötigte Werkzeug miteinander verbindet“. Weiterhin muss es „[. . .]auf die Aufgabe, Fähigkeiten, Erfahrungen und Präferenzen des Benutzers sowie die Funktionalität des Werkzeugs abgestimmt sein“ [31]. Symboliken, die der Benutzer aus seinem Alltag kennt und einzusetzen weiß, helfen ihm bei der Bedienung neuer Komponenten. Daher werden zur Begrenzung des zeitlichen Intervalles des Sliders Objekte in Form eines Dreiecks eingesetzt. Mit dieser Darstellung wird eine Assoziation zu einer Pfeilspitze hergestellt, was die Einordnung der aktuellen Jahreszahlen erleichert. Anhand einer dezenten dreidimensionalen Erscheinung sowie einer Schattierung wirkt die Komponente wie ein bewegbarer Slider mit zwei Thumbs. Abbildung 4.11 zeigt wie sich die Position der Pfeilkomponenten auf den Slider der Tabelle auswirkt. In der Abbildung ist ebenfalls die Hervorhebung eines berührten Pfeils zu sehen. Die Größe der pfeilförmigen Objekte richtet sich nach der Mindestgröße für alle Schaltflächen der Anwendung von 30 mal 30 Pixeln. Jedoch weist die Dreiecksform deutlich weniger berührbare Fläche auf, als die quadratischen Flex-Buttons. Deshalb wird zusätzlich ein für den Benutzer nicht sichtbarer Bereich um die Pfeile gelegt, der die anfassbare Fläche um 5 Pixel in Höhe und Breite vergrößert. Damit erleichtert sich die Auswahl dieser Bedienelemente, was zu einer verbesserten Benutzerfreundlichkeit führt.

Abbildung 4.11: Mit den bewegbaren Pfeilen kann der Benutzer die angezeigten Jahreszahlen beeinflussen

Checkbox

Die Checkboxen müssen für eine Verwendung auf Touchscreens hinsichtlich ihrer Größe geändert werden. Dabei wird ein Item-Renderer für alle Checkboxen eingesetzt. Abbildung 4.12 zeigt die verhältnismäßige Änderung zu den bisherigen Checkboxen. Dabei ist ersichtlich, dass sowohl die Checkboxen der Tabellenzeilen als auch die Box aus dem Header den neuen Skin annehmen.

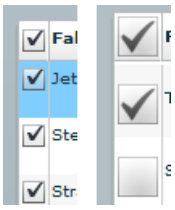

Abbildung 4.12: Links - die standardmäßigen Checkboxen, rechts - die auf Fingergröße optimierten Komponenten

Textfelder

Die Textfelder hingegen ändern sich nur für die Zeilen, die gerade gewählt sind. Dabei heben, wie in Abbildung 4.13 zu erkennen, eine höhere Schriftgröße und Gewichtung die aktuellen Textfelder hervor.

Abbildung 4.13: Links - die Textfelder im nicht selektierten Zustand, rechts - die Schriften ändern sich mit Auswahl einer Zeile

Wird ein Textfeld in den Bearbeitungsmodus versetzt, erscheint eine TextArea-Komponente mit den standardmäßigen Eigenschaften für die Schrift. Abbildung 4.14 zeigt ein bearbeitbares Regionen-Textfeld.

Abbildung 4.14: Ein Textfeld im bearbeitbaren Zustand hebt sich deutlich von den umgebenden Feldern ab

4.3 Zeitstrahl

Der Bereich Zeitstrahl betrifft die Slider-Komponente, die als Item-Renderer wirkend nur einmal als Klasse definiert werden muss und für jede sichtbare Zeile in der Tabelle neu instanziiert wird. Die Komponente zeichnet sich durch mindestens zwei Symbole oder

auch Thumbs aus, die bislang mit der Maus verschoben werden konnten. Zusätzlich finden sich in dem Zeitstrahl über jedem Thumb Informationen zu dessen zeitlicher Einordnung. Dieser Bereich des *Auswahl*-Fensters bietet mit seinen Funktionalitäten viel Potenzial für neue auf berührungsempfindliche Geräte ausgelegte Interaktionsmöglichkeiten.

4.3.1 Interaktion

Halbkreismenü aus Adobe Flash Professional

Die bisherigen Funktionen zum Manipulieren der Datensätze fielen entweder durch viele Formulare oder dem Festhalten einer Taste auf. Beide Möglichkeiten eignen sich nicht für die Interaktion mit einem Touch-Gerät. Daher muss überdacht werden, inwiefern einzelne Funktionen ersetzt werden können. Es sind spezielle Gesten denkbar, welche die Aufgaben der bisherigen Funktionen ablösen könnten. Damit wäre zumindest die Unabhängigkeit von der Tastatur gewährleistet. Allerdings bedeutet dies auch, dass sich unerfahrene Benutzer diese Gesten erst aneignen müssten. Dadurch wäre vor allem die intuitive Bedienung der Komponenten in Frage zu stellen. Andererseits kann die Bedienung des Zeitstrahls mit Hilfe eines zusätzlichen Aktionenmenüs erleichtert werden. Dabei ist es vorgesehen, dass bei einer Berührung der Slider-Komponente ein Menü in Form eines Halbkreisausschnittes über der Komponente erscheint. Ein Grund für den Einsatz eines solchen Menüs, ist die übersichtliche und gekapselte Darstellung von Symbolen, die der Benutzer als Schaltflächen wahrnehmen soll. Weiterhin eignet sich diese Form der Menüdarstellung sehr gut, um ein Menü unmittelbar im zentralen Bereich der Anwendung einzusetzen, ohne überladend und ablenkend zu wirken. Der Bezug zwischen der aktiven Zeile und dem Benutzer bleibt ebenfalls erhalten. Des Weiteren bringt die grafische Form mehr Dynamik in die Anwendung und zeigt dem Benutzer sofort an, dass er für die gewünschten Funktionsaufrufe mit dem Menü arbeiten muss.

Die Abbildung 4.15 zeigt, dass sich in diesem Menü bislang eine aktive Schaltfläche befindet. Ihr dreidimensionaler Stil gibt dem Benutzer zu verstehen, dass er sie berühren kann. Die Schaltfläche besteht aus einem einfachen Logo, das einer bestimmten Funktionalität zugeordnet wird. Mit dem Auflegen des Fingers auf die Schaltfläche oder mit dem Wischen darüber wird eine Funktion aufgerufen und das Aktionenmenü ausgeblendet. Berührt der Benutzer bei angezeigtem Aktionenmenü eine andere Stelle von Cyber, verschwindet das Menü ebenso. Alle Funktionen, die in dem Menü aufgerufen werden, haben nur Einfluss auf die entsprechende Slider-Komponente in der Zeile, in der das Menü in Erscheinung tritt.

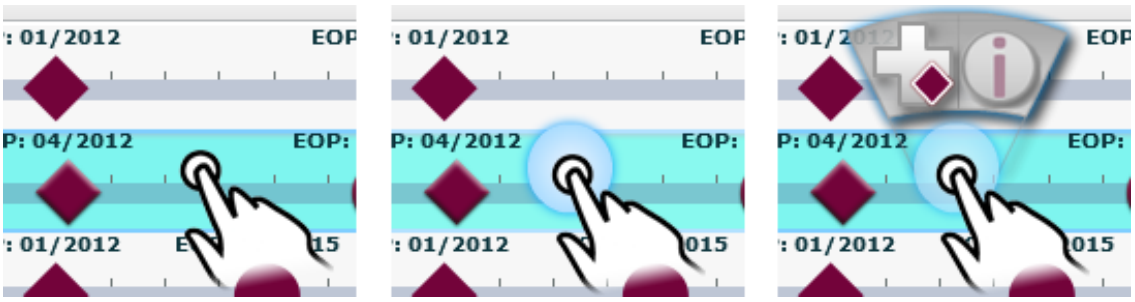

Abbildung 4.15: Das Menü baut sich nach Auflegen des Fingers durch zwei Zustände auf (vgl. Quelle: [18])

Hinzufügen einer GP mit einem Halbkreismenü

Das Menü wird beim Hinzufügen einer GP zum Einsatz kommen. Legt der Benutzer einen Finger eine Sekunde auf eine leere Stelle des Slider, erscheint das Menü. Es gibt dem Nutzer zu verstehen, dass er an der berührten Stelle eine GP hinzufügen kann. Berührt er die Schaltfläche im Menü, wird die entsprechende Funktion zum Eintragen der neuen GP aufgerufen. Daraufhin kann der aktuelle Vorgang bestätigt oder abgebrochen werden. Damit es nicht möglich ist, eine GP vor einem SOP oder nach einer EOP einzutragen, wird die mögliche Interaktionsfläche begrenzt. So kann nur in dem Slider zwischen SOP und EOP das Menü aufgerufen werden. Zur einfacheren Darstellung dieser Begrenzung wird eine hellgrüne Fläche den SOP mit dem EOP eines Sliders verbinden und dem Benutzer damit zu verstehen geben, dass nur dort ein aufgelegter Finger das GP-Menü aufrufen kann.

Löschen einer GP mit einer Geste

Damit auch das Löschen einer GP möglichst einfach vollzogen werden kann, wird diese Funktionalität dem Benutzer in einer anderen Form als bisher angeboten. Eine Möglichkeit wäre, auch hierfür das Halbkreismenü wieder einzublenden. Dies hätte zu Überschneidungen mit dem Aufrufen des Menüs zum Hinzufügen geführt und kann deshalb nicht eingesetzt werden. Somit erscheint das Menü nur beim Hinzufügen einer GP. Soll eine bestehende GP gelöscht werden, muss der Finger auf eine der GP-Grafiken gelegt werden und diese mit einer Drag-Geste nach oben oder unten verschoben beziehungsweise geschossen werden. Die Abbildung 4.16 verdeutlicht die vertikale Bewegung. Diese kommt einem Wegziehen aus dem Zeitstrahl gleich und damit auch dem Entfernen aus der Liste.

Single-Drag And Drop

Mit Multitouch als Interaktionsform überzeugt nicht nur die „[...] hohe visuelle und ästhetische Qualtität der Benutzungsschnittstellen, sondern auch die Direktheit der Inter-

Abbildung 4.16: Wird eine GP aus ihrer Zeile bewegt, kann sie gelöscht werden (vgl. Quelle: [18])

aktion“ [12]. Grafische Objekte lassen sich direkt mit dem Finger bearbeiten, was mit externen Geräten wie der Maus nicht auf diese Art möglich ist. Um den Slider unmittelbar bedienen zu können, wurde sich um möglichst intuitive Interaktionsmöglichkeiten bemüht. Daher soll die Verschiebung einzelner SOPs, EOPs oder GPs wie in der Ausgangsversion mit einem Drag And Drop realisiert werden. Der Benutzer kann damit den Finger auf eines der Objekte legen und es verschieben. Wird die Berührung unterbrochen, bleibt das Objekt an der Position des Fingers und die neuen Projektinformationen werden gespeichert. Die Abbildung 4.17 verdeutlicht das Prinzip. Dieser Vorgang ist vielen Nutzern im Vornhinein bekannt und wird daher als äußerst intuitiv bewertet.

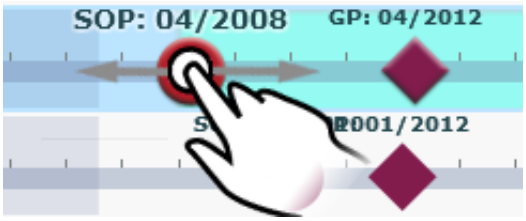

Abbildung 4.17: Einzelne Slider-Thumbs sind mit einem Finger bewegbar (vgl. Quelle: [18])

Multi-Drag And Drop

Eine weitere Funktionalität des ursprünglichen Sliders war das komplette Versetzen aller Produktionsabschnitte auf der Zeitleiste des Sliders. Dies wird auf Touchscreens durch das gleichzeitige, parallele Verschieben von SOP und EOP möglich sein. Bei dieser Zwei-Finger-Geste wird ein Finger auf den SOP und ein weiterer auf den EOP gelegt. Werden nun beide möglichst parallel zur selben Zeit verschoben, bewegen sich die zugehörigen GPs demenentsprechend. Die Abbildung 4.18 verdeutlicht diese Geste, bei der sich in diesem Beispiel der SOP und GP an der Position des EOP orientieren. Sobald beide Finger die Oberfläche wieder verlassen, erscheint ein Fenster, in dem die Änderungen abschließend bestätigt oder verworfen werden können.

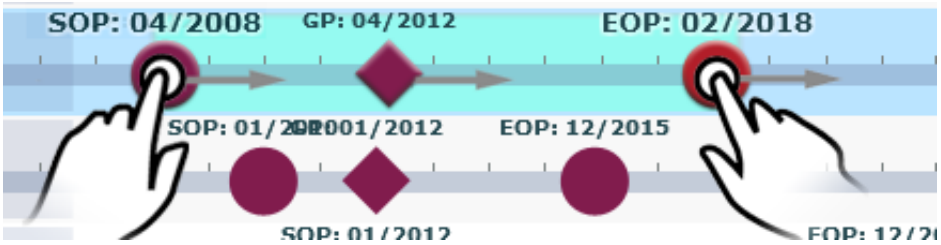

Abbildung 4.18: Die Multi-Drag-Geste dient zum gleichzeitigen, parallelen Verschieben aller Projektabschnitte (vgl. Quelle [18]

Slider-Stil

Ein Kritikpunkt bei der Analyse der ursprünglichen Cyber-Anwendung betraf die fehlende visuelle Reaktion des Systems auf Eingaben oder Bewegungen. Auch die berührbaren Elemente wurden wenig hervorgehoben und wirkten oft nicht greifbar. Beide Problematiken werden in dem Entwurf des Sliders berücksichtigt. So ändert der Slider seine Erscheinung, wenn die zugehörige Zeile ausgewählt ist. Neben einer Hervorhebung der Drag And Drop-Funktionalität anhand dreidimensionaler Objektgrafiken erscheint ein heller Bereich zwischen dem jeweiligen SOP und EOP. Dieser Bereich deutet die mögliche Fläche an, in der eine GP von dem Benutzer hinzugefügt werden kann. Außerdem müssen für eine Interaktion mit dem Finger die Rauten- und Kreisgrafiken der Slider-Komponente vergrößert werden.

Zur besseren Übersicht werden die Zusatzinformationen in Form von Labels zu den einzelnen Produktionsabschnitten anders positioniert. Ansonsten besteht ein hohe Risiko, dass bei der Verschiebung eines Punktes die Labels von dem Finger verdeckt werden. Dadurch wäre eine genaue Positionierung des Punktes anhand der zeitlichen Einordnung nicht gewährleistet. Demzufolge treten die Zusatzinformationen eines aktiven Thumbs hinsichtlich ihrer Positionierung und Schriftgröße nun stärker hervor. Um die Hervorhebung der Labels auch optisch besser wirken zu lassen, erfolgt sie anhand einer Animation. Sowohl die Schriftgröße als auch die Umrandung des Textes ändern sich dadurch mit der Auswahl eines Thumbs flüssig.

4.3.2 Software

Halbkreismenü aus Adobe Flash

Das Menü in Form eines Kreisausschnittes wurde in Adobe Flash Professional CS4 erstellt. Durch die Funktion zum Export als Flex-Komponente konnte dieses Element als Shockwave Flash Component (SWC)-Datei[21] in das Flex-Projekt eingebunden werden. Dadurch wird voller Zugriff auf alle in Flash erstellten MovieClips und Tweens ermög-

[21] Eine SWC-Datei speichert alle zur Ausführung des Flash-Filmes benötigten Komponenten.

licht. Für verschiedene Sequenzen, wie dem Einblenden des Menüs, wurden in Flash entsprechende Tweens mit eindeutigen Bildbezeichnungen erzeugt. Anhand dieser Bezeichnungen kann im Flex-Projekt von einem Zustand zu einem anderem geschalten werden. Damit sich die Zustände erst nach einer kurzen Pause ändern, werden die zugehörigen Funktionen von voreingestellten Timern aufgerufen. Flex verwendet dabei automatisch den in Flash eingestellten Tween, was eine ansehnliche und mit verschiedenen Filtern versehene Bewegung zur Folge hat.

Hinzufügen einer GP mit einem Halbkreismenü

Damit das Menü erscheint, werden mit dem Auftreten eines TouchBegin-Ereignisses die aktuelle Zeile und das betreffende Ziel abgefragt. Sollte die Zeile bereits aktiv und die berührte Komponente kein SOP, EOP oder GP sein, wird an der betreffenden Stelle eine neue Instanz des Halbkreismenüs erzeugt. Mit Timern verzögert, erscheint zunächst ein blauer Kreis als Markierung des Berührungspunktes. Nach dieser Verzögerung blendet auch das restliche Menü auf. Dort kann der MovieClip aus dem ehemaligen Flash-Menü direkt angesprochen werden und reagiert auf TouchTap- und Touch-Roll-Over-Ereignisse. Sobald das Menü wieder ausblenden soll, wird der entsprechende Zustand gesetzt. Dadurch folgt die Applikation wieder dem in Flash eingestellten Tween, blendet das Menü aus und löscht die aktuelle Instanz von der DisplayList.

Löschen einer GP mit einer Geste

Die Programmierung solch einer Geste zum Wegstoßen eines Objektes ist mit den Touch-Events ohne großen Aufwand verbunden. Über die Eigenschaft *stageY* des Zielobjektes kann in der TouchMove-Methode verglichen werden, wie weit sich der aufgelegte Finger von seinem Ausgangspunkt bewegt hat. Wird ein bestimmter Grenzwert überschritten, blendet das Objekt mit einer Animation aus und ruft die Funktion zum Bestätigen des Löschvorganges auf. Die Animation erfolgt anhand eines Spark-Effekts namens *Fade*. Mit diesem vorgefertigten Effekt kann nach Zuweisung des Zielobjektes und den Alphawerten auf einfache Weise ein ansehnlicher Ausblendeffekt eingestellt werden.

Single-Drag And Drop

Um die intuitive Drag And Drop-Funktionalität der Slider-Thumbs zu erhalten, bedarf es keiner Umstellung. In diesem Fall werden standardmäßig ohne spezielle Event-Listener die Berührungen als Mouse-Events interpretiert, mit denen Slider-Thumbs ohne Weiteres verschiebbar sind. Deshalb müssen für diese Funktionalität keine *StartTouchDrag*-Funktionen oder ähnliche Anweisungen geschrieben werden.

Multi-Drag And Drop

Das Flex-Framework erkennt mit seiner Multitouch-unterstützenden API zwar zahlreiche auftretende Berührungspunkte und damit auch Gesten. Allerdings kann nur ein grafisches Element im Normalfall mit dem Benutzer interagieren. Wenn also ein Finger auf einer DataGrid aufliegt, reagiert Flex darauf und wählt eine Zeile aus. Bleibt der Finger auf der Oberfläche und berührt ein zweiter eine andere Komponente, wird diese Aktion von der Anwendung ignoriert. Dieses Beispiel kann auf nahezu alle grafischen Elemente in Flex übertragen werden. Um dem entgegenzuwirken bietet das Framework an, bei Auftreten eines TouchBegin-Ereignisses darauf zu reagieren. Zumindest bei Drag And Drop-Funktionen funktioniert dieses Verhalten. Dabei müssen nur die Funktionen zu *StartTouchDrag* und *StopTouchDrag* aufgerufen werden.

Dies konnte bei den für die Zoom-Funktion nötigen Multi-Drag-fähigen Pfeilkomponenten aus 4.2.2 *Software* erreicht werden, ist aber nicht ohne Weiteres in einer Slider-Komponente anwendbar. Jeweils ein Thumb des Sliders ist auch ohne Beginn eines TouchDrags verschiebbar. Wenn trotzdem in jedem Thumb bei Beginn eines Touch-Events die *StartTouchDrag*-Methode aufgerufen wird, sind zwar alle Thumbs verschiebbar, aber ihre Positionen stimmen nicht mit den Fingerpositionen überein. Lediglich die Lage des zuerst berührten und verschobenen Objektes ist korrekt. Wird ein zweiter Thumb gleichzeitig bewegt, ist kein sauberer Bewegungsfluss erkennbar. Häufige Sprünge nach vorn und zurück ermöglichen keine korrekte Bedienung eines zweiten oder auch dritten Thumbs. Dieses Verhalten entsteht dadurch, dass die *StartTouch-Drag*-Methode bei dem ersten Objekt ignoriert wird, da der Slider standardmäßig eine Verschiebung dieses Objektes zulässt. Wird ein zweites berührt, so wird ein Single-Drag And Drop nicht mehr aufgerufen. Dadurch greift das Objekt auf die Touch-Drag-Methode zurück, welche in einer Slider-Komponente zu großen Problemen führt.

Damit trotzdem eine dem Multi-Drag And Drop ähnliche Geste realisiert werden kann, wird der Single-Drag And Drop erweitert. Dafür müssen die Zielobjekte in den Touch-Move-Methoden bei einer festen Anzahl von zwei Fingerberührungen dem SOP und dem EOP entsprechen. Diese Methoden werden nicht nur durch Bewegung, sondern auch durch das einfache Auflegen des Fingers aufgerufen. Wird der zuerst berührte Thumb bewegt, verschieben sich alle anderen Objekte ebenfalls. Allerdings wird deren Position aus dem Verhältnis zu dem verschobenen und rot hervorgehobenen Thumb kalkuliert. Es entsteht also nur der Eindruck, der Benutzer könne mit zwei Fingern alle Thumbs verschieben. Das eindeutige Problem dabei ist, dass nicht-parallele Bewegungen ignoriert werden.

Slider-Stil

Da der Slider wie schon die Checkbox als Item-Renderer eingesetzt wird, müssen hierbei die den Item-Renderer betreffenden Dateien geändert werden. Um auf die Grafiken

der Thumbs zuzugreifen, wird in der Slider-Komponente die *styleName*-Eigenschaft des Thumbs eingestellt. Abhängig von der aktuellen Zeilenauswahl erscheint ein Thumb zwei- oder dreidimensional. Für diese Einstellung finden die *isItemSelected*- und *isItemHighlighted*-Eigenschaften des ListData-Objektes ihren Einsatz. Die Grafiken für alle Thumbs werden in einem Style-Tag über Eigenschaften wie *ThumbUpSkin* oder *ThumbOverSkin* bereits für die zwei verschiedenen Status eingebettet.

Der Bereich, der es ermöglicht, GPs nur in einer begrenzten Fläche hinzuzufügen, wird durch eine BorderContainer-Komponente realisiert. Damit die Größe und Positionierung dieses Containers für jede Instanz des Slider-Item-Renders stimmen, errechnen sich diese Werte aus den übergebenen Werten von SOP und der aktuell angezeigten Jahresanzahl. Sowohl die Breite als auch Anfangs- und Endpunkte passen dadurch immer zu dem betreffenden SOP und EOP.

Um die Labels des Sliders zu ändern, wird die Slider-Thumb-Komponente angepasst. Die dort erzeugten Labels werden über dem Thumb positioniert und bekommen einen Text zugewiesen. Wird ein Thumb berührt, ändern sich Umrandung und Schriftgröße der Labels. Umrandungen können in dem Flex 4-Framework mit dem *Glow*-Effekt erzeugt werden. Für Animationen von Stileigenschaften wie der Schriftgröße des Textes bedarf es hingegen der Verwendung der Klasse *Animate Property*. Durch Zuweisung des Zielobjektes sowie Ausgangs- und Endwerten für den Schriftstil kann dadurch auf einfache Weise eine flüssige Änderung erwirkt werden. Damit beide Animationen auch gleichzeitig ablaufen, bietet sich der Einsatz eines *Parallel*-Objektes an. Wie ein Container kann es die beiden Effekte aufnehmen und gleichzeitig abspielen.

4.3.3 Screendesign

Slider

Damit die Bindung zwischen dem Benutzer und der Anwendung erhöht wird, reagiert auch die äußerliche Erscheinung des Sliders auf Berührungen. So werden die Thumbs eines Sliders wie in Abbildung 4.19 dreidimensional dargestellt, wenn dessen Zeile aktiv ist. Dreidimensionale Objekte verdeutlichen stärker, dass diese greifbar sind und in diesem Fall auch verschoben werden können. Der Benutzer merkt dadurch schneller, in welcher Form er mit diesen Objekten agieren kann, was die Benutzerfreundlichkeit merklich verbessert. Wird eine Zeile wieder abgewählt, erscheinen die Thumbs zweidimensional. In diesem Zustand können die Objekte nicht verschoben werden.

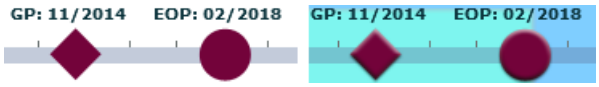

Abbildung 4.19: Links - der Slider im inaktiven Zustand, rechts - aktiver Slider mit Hervorhebung der Thumbs

Selbst wenn ein Thumb berührt wird, erhält der Benutzer eine visuelle Reaktion von der Anwendung. Zum Einen färbt sich die Oberfläche eines aktiven Thumbs rot. Zum Anderen vergrößert sich das zugehörige Label. Diese Änderung nimmt der Benutzer als Animation von einer kleinen Schriftgröße zu einer größeren wahr. Abbildung 4.20 zeigt, dass dem Label zur besseren Hervorhebung außerdem ein weißer Rahmen beigefügt wird.

Abbildung 4.20: Links - ein berührter GP, rechts - ein hervorgehobener EOP

Kreisausschnittmenü

Das Menü zum Hinzufügen einer weiteren GP muss für den unerfahrenen Benutzer selbsterklärend und intuitiv bedienbar sein. Es besteht aus einer Schaltfläche zum Aufruf der Hinzufügen-Funktion, einem Menühintergrund und einer kreisförmigen Grafik, die den aktuellen Berührungspunkt darstellt. Abbildung 4.21 zeigt, dass das Menü noch Platz für eine weitere Schaltfläche bietet. Dieser Bereich wird bislang von einem ausgegrauten Platzhalter belegt. Bei Weiterentwicklungen dieser Anwendung kann das Menü dadurch um weitere Funktionalitäten wie den direkten Aufruf weiterer Projektinformationen ergänzt werden.

Abbildung 4.21: Das Menü zum Hinzufügen einer weiteren GP

Damit ein Berührungspunkt auch visuell dargestellt werden kann, blendet nach dem Auflegen eines Fingers ein hellblauer Kreis ein. Ein weicher Farbverlauf sowie die dezente Farbgebung heben den Berührungspunkt zwar hervor, lassen ihn aber noch als ein Teil der Anwendung wirken. Die Abbildung 4.22 zeigt einen beispielhaften visuellen Berührungspunkt.

Die Schaltfläche des Menüs zeichnet sich durch ein eindeutiges Symbol aus. Das Plus-Zeichen signalisiert dem Benutzer, dass ein Objekt hinzugefügt werden kann. Die Raute gibt zu verstehen, dass es sich bei dem Objekt um eine GP handelt. Der Benutzer kennt das Symbol der Raute bereits von den Thumbs der Slider-Komponente. Deshalb fällt

Abbildung 4.22: Bei Kontakt mit dem Slider erscheint zuerst ein Kreis zur Hervorhebung des Berührungspunktes

die Einordnung der aufrufbaren Funktion leichter. Da das Hinzufügen dabei die oberste Priorität besitzt, wird das Plus wie in Abbildung 4.23 erkennbar, um ein Vielfaches größer dargestellt als die Raute.

Abbildung 4.23: Das Symbol zum Hinzufügen einer GP

4.4 Fazit

In dem Kapitel *Konzept* wurden drei Komponenten definiert, mit denen es nach der Implementierung in Cyber möglich sein soll, das *Auswahl*-Fenster auf einem berührungsempfindlichen Gerät zu bedienen. Jede dieser Komponenten zeichnet sich sowohl durch grafische Elemente als auch durch zusätzliche Funktionalitäten aus. Wichtig für eine optimale Bedienung dieser Elemente ist die Festlegung einer allgemeinen Mindestgröße. Anhand einiger ausgewählter Richtlinien müssen die zur Interaktion mit dem menschlichen Finger ausgelegten Komponenten mindestens 30 Pixel breit und hoch sein. Damit der Benutzer nicht nur Elemente verschieben und auswählen kann, sondern auch mitgeteilt bekommt, wenn er etwas berührt, reagiert die Anwendung visuell auf Eingaben. Somit ändert sich das Interfacedesign am Beispiel von Tabellenzeilen und Slider-Thumbs bei deren Berührung.

Auch der Einsatz von Gesten wurde geprüft. Der Nutzer wird künftig mit Ein-Finger-Gesten wie DoubleTap aber auch Zwei-Finger-Gesten zum Scrollen und MultiDrag die Anwendung bedienen können. Genauso wurde auch über Alternativen zu aufwendigeren Gesten nachgedacht. Es wird zwar eine Zoom-Funktion in der Cyber-Anwendung implementiert, diese soll aber nicht als Geste Verwendung finden. Als Ersatz werden Slider-ähnliche Pfeilkomponenten gleichzeitig verschiebbar sein, welche aufgrund ihrer Erscheinung und Bedienung als intuitiver sowie weniger komplex einzuschätzen sind als die Zoom-Geste. Außerdem wird mit der grafischen Einbindung in die Benutzeroberfläche das Risiko verringert, dass der Benutzer nichts von der Existenz einer Zoom-Funktion weiß.

Für die Umsetzung der grafischen Komponenten sowie der Funktionalitäten wurden Ansätze erläutert. Dabei standen vor allem die Verwendung der Touch-Events im Vergleich zu den Flex-internen Gesten, aber auch Item-Renderer sowie die Einbindung externer Projekte wie ObjectHandles im Mittelpunkt. Es wurde erkannt, dass die Multitouch-API, die das Flex-SDK bereitstellt, nur begrenzt einsetzbar ist. Keine plattformübergreifende Gestenunterstützung sowie der nicht kombinierbare Einsatz von Touch- und Gesture-Events ließen nur die Erstellung eigener Gesten zu.

5 Implementierung

In diesem Kapitel wird die Implementierung der in dem Konzept definierten Eigenschaften und Funktionalitäten der Komponenten erläutert. Dabei werden im Vorfeld der Aufbau des Flex-Projektes sowie der Zusammenhang der für diese Arbeit relevanten Klassen dargestellt. In einem weiteren Schritt wird die Implementierung der in dem Konzept definierten Kompontenen in das Cyber-Projekt erläutert. Hierbei erfolgt eine Teilung des *Auswahl*-Fensters in die drei bekannten Bereiche Fenster, Tabelle und Zeitstrahl. Damit wird eine bessere Zuordnung zu den Software-Ansätzen aus dem Konzept ersichtlich. Anhand einiger Quellcodeauszüge wird die Umsetzung der wichtigsten Funktionalitäten erklärt. Dabei rücken vor allem die Verwendung der Touch-Events, der Einsatz von Item-Rendern sowie die Implementierung externer Klassen wie ObjectHandles in den Mittelpunkt

5.1 Projektaufbau

Um den Aufbau des Cyber-Projektes zu veranschaulichen, stellt die Abbildung 5.1 das Flex-Projekt in seinen höchsten Stufen dar. Anhand einer Paketstruktur werden dabei die obersten Paketschichten abgebildet. Jedes dieser Pakete steht dabei für eine wesentliche Funktionalität oder für einen Typ von Komponenten.

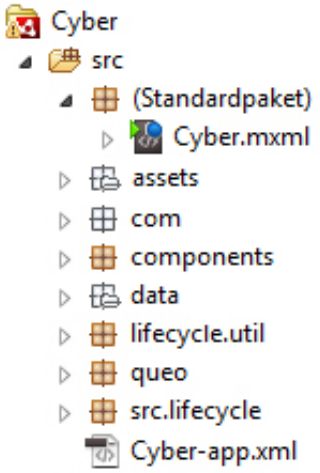

Abbildung 5.1: Die Ordnerstruktur des Cyber-Projektes im Flex Framework

Die in der Entwicklungsumgebung kompilierte Haupt-Datei stellt dabei die *Cyber.mxml* dar. Diese wird bei jedem Kompilierungsvorgang ausgeführt und greift auf alle weiteren Ordner und Unterordner zu. Wesentliche Einstellungen zu Größe, Positionierung oder Namen der am Ende erstellten AIR-Datei können in der am untersten Ende der Struktur befindlichen und automatisch generierten *Cyber-app.xml* vorgenommen werden. Dabei sind Werte wie die Initialisierungsgröße oder die Anfangsposition der Anwendung

festlegbar. Da Cyber im Vollbildmodus aktiv sein soll, werden diese Eigenschaften ignoriert.

Alle wesentlichen grafischen Komponenten des Projektes befinden sich in dem *components*-Paket, wobei hier nur die MXML-Komponenten berücksichtigt werden. Diejenigen Elemente, die von benutzerdefinierten Action Script-Klassen erben, wurden dem *com*-Paket zugeteilt. Dazu gehören weitere Klassen zu speziellen Events, Dialogfenstern oder Ausnahmeregelungen. In dem *queo*-Paket findet man weitere, individuelle Komponenten sowie einen großen Teil der eingebetteten und zur Laufzeit geladenen Grafiken. Zusätzliche grafische Objekte befinden sich in dem Ordnet *assets*. Das *styles*-Paket dient zum Festlegen bestimmter Stileigenschaften in CSS-Dateien. Ein wichtiger Bestandteil des gesamten Projektes hinsichtlich der Kommunikation mit dem Back-End ist das *src.lifecycle*-Paket. Mit all seinen Klassen gewährleistet es den festgelegten Ablauf des Projekt-Lifecycles. Neben Anfragen an das Back-End spielen dabei auch Objektmodelle, Event-Handler und vordefinierte Datenobjekte eine wichtige Rolle.

5.2 Klassenüberblick

In diesem Abschnitt wird erläutert, welche Klassen für die Implementierung der Multitouch-Komponenten in das bestehende Flex-Projekt implementiert wurden. Ausgangspunkt ist dabei die *Cyber*-Klasse, die weitere Komponenten wie Panel oder DividedBox aufnimmt. Dort wird auch die MXML-Datei *Table* einmalig verwendet, die das *Auswahl*-Fenster realisiert. Abbildung 5.2 zeigt, dass *Table* zahlreiche weitere MXML-Komponenten integriert und von der Klasse *QueoListDataGrid* erbt.

Das Klassendiagramm zeigt nur einen Ausschnitt der in Cyber verwendeten Klassen. Die hier ausgewählten Elemente besitzen aber für diese Arbeit hohe Priorität und wurden entweder für diese erweitert oder neu erstellt. Wichtige Funktionen und Eigenschaften der Klassen wurden zur besseren Einordnung mit hervorgehoben. Die *Table*-Klasse erbt nicht nur, sondern hat auch Zugriff auf weitere Klassen, die in einigen Fällen als Item-Renderer eingesetzt werden. Mit der MXML-Komponente *SliderIntervalComp* wird das Element rund um die bewegbaren Pfeilkomponenten zum Manipulieren des Intervalls implementiert. Auf die grafischen Objekte sowie die Logik rund um den Drag And Drop-Prozess kann die Komponente durch die *SliderInterval*-Klasse zugreifen. Um in der Tabelle das GP-Kreismenü anzuzeigen, wird in *Table* eine neue Instanz von *GPmenu* erstellt. Diese Klasse hat vollen Zugriff auf die in Flash Professional erstellte und exportierte *gpMenuLib.swc*-Datei, die als Bibliothek in das Flex-Projekt importiert wurde.

Weiterhin werden drei Renderer-Klassen in die Tabelle eingefügt, die in der DataGrid als Item-Renderer zum Einsatz kommen. Neben den in Abbildung 5.2 aufgelisteten drei Komponenten wird für jede dieser Komponente noch ein sogenannter Header-Renderer

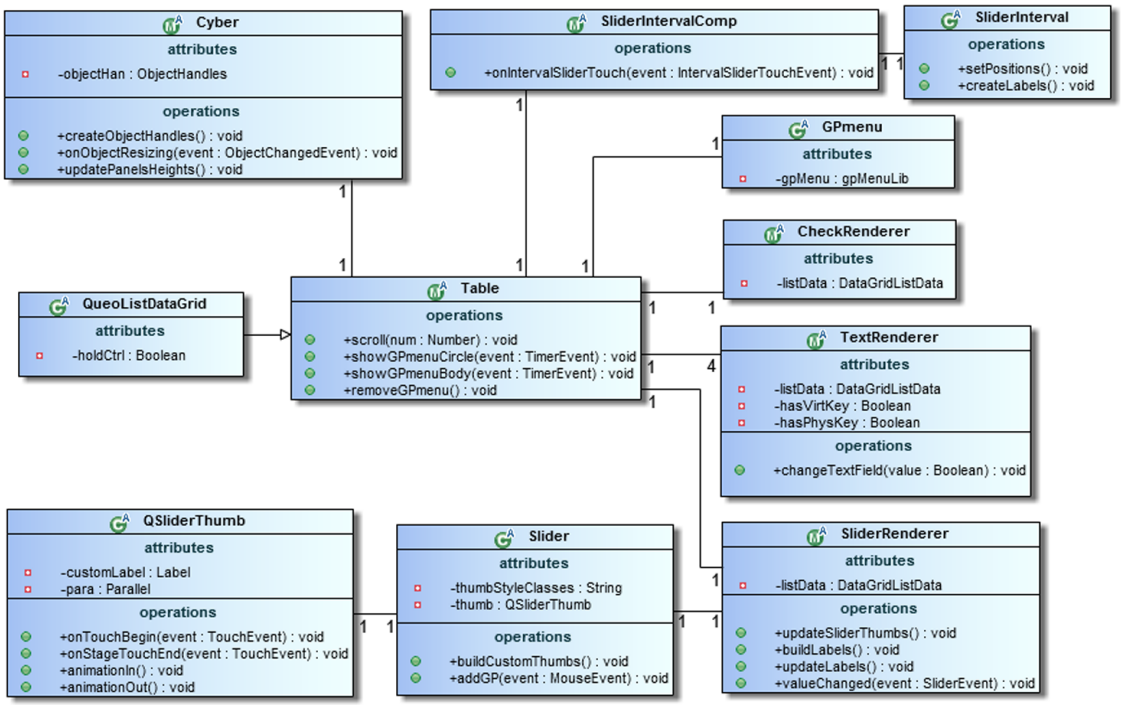

Abbildung 5.2: Das UML-Diagramm stellt die Verbindung der wichtigsten Klassen dieser Arbeit dar

verwendet. Dieser wurde der besseren Übersicht wegen in der Übersicht vernachlässigt. Mit der *CheckRenderer*-Klasse wird die in der ersten Spalte berührungsempfindliche Checkbox realisiert. Diese muss, wie auch alle weiteren Renderer, für den Zugriff auf die Tabellendaten auf das DataGridListData-Objekt zurückgreifen. Der *TextRenderer* dient zum Anzeigen und Bearbeiten der Textfelder in der Tabelle. Diese Komponente wird viermal in der DataGrid als Renderer angewendet. Zum Implementieren des Sliders wurde die Klasse *SliderRenderer* eingesetzt. Diese erhält Zugriff auf die Label- und Stilfunktionen der Action Script-Klasse *Slider*. Wichtig für die Erstellung eines benutzerdefinierten Slider-Thumbs ist die Klasse *QSliderThumb*. Mit ihr werden alle Funktionen zur Animation des Thumbs und der Labels sowie zur Erkennung und Verarbeitung der Touch-Events implementiert.

5.3 Implementierung der Komponenten

In den folgenden Abschnitten wird die Implementierung der in dem 4. Kapitel *Konzept* definierten Komponenten erläutert. Dabei sollen weniger die Funktionen zur Projektdatenimplementierung im Vordergrund stehen. Der Schwerpunkt liegt auf der Verwendung der Flash-internen Touch-Events sowie der Anpassung, der mit dem Flex-Framework angebotenen, grafischen Komponenten. Neben den standardmäßigen Flex-Steuerungselementen soll auch die Implementierung individueller Komponenten Erwähnung finden.

5.3.1 Fenster

Anfasser

Um die Anfasser in die Anwendung zu implementieren bedarf es zunächst einer Erstellung eines neuen ObjectHandles-Objektes. Ihm wird bei der Instanziierung mitgeteilt, auf welchen Container es sich auswirken soll. In diesem Fall wird es auf den Container *tablebox* angewendet, der die DataGrid aufgenommen hat. Weiterhin wird mit *new ClassFactory* darüber entschieden, welche Form die Anfasser haben sollen. Dabei wurde sich auf einen Kreis festgelegt, der in der *CircleHandle*-Klasse näher definiert wurde. Als abschließende Variable wird ein neuer *Flex4ChildManager* erzeugt, der dem Objekt mitteilt, dass es sich bei der Anwendung um ein Flex 4-Projekt handelt. Dadurch können Probleme beim Hinzufügen und Löschen von Kindelementen umgangen werden.

```
//Klasse Cyber.mxml
private function createObjectHandles():void{
    //Erzeugung einer neuen ObjectHandles-Instanz
    objectHan = new ObjectHandles(tableBox, null, new
       ClassFactory(CircleHandle), new Flex4ChildManager());
    //Festlegen weiterer Eigenschaften
}
```

Damit die Größe des verschiebbaren Bereiches definiert wird, muss ein neues *SizeConstraint*-Objekt instanziiert werden. Diese Klasse ist ein Teil der ObjectHandles-Bibliothek. Minimale und maximale Werte können nach dem Schema *SizeConstraint.Begrenzungseigenschaft* eingestellt und abschließend dem ObjectHandles-Objekt zugewiesen werden.

```
//in Funktion createObjectHandles
//Beschränkung des verschiebbaren Bereiches
var constraint:SizeConstraint = new SizeConstraint();
constraint.minWidth = 500;
constraint.minHeight = 200;
objectHan.addDefaultConstraint(constraint);
```

Standardmäßig können bei einem ObjectHandles-Objekt alle vier Anfasser bedient und gezogen werden. Im Fall von Cyber sollen aber der linke, obere Anfasser nicht und zwei weitere nur begrenzt verschiebbar sein. Für dieses Szenario kann mit einem Array namens *handles* für jeden Punkt die mögliche Bewegungsrichtung angegeben werden. In dem folgenden Codeauszug wird dem oberen, rechten Punkt mitgeteilt, dass er aufgrund von *RESIZE_RIGHT* nur nach rechts verschiebbar sein soll. Diese Anweisung wird dem Array hinzugefügt, wodurch es nach einer weiteren Zuweisung für das ObjectHandles-Objekt gilt.

```
//in Funktion createObjectHandles
//Bewegungsrichtungen festlegen
var handles:Array = [];
```

```
handles.push(new HandleDescription(HandleRoles.RESIZE_RIGHT,
   new Point(100,0), new Point(0,0)));
//weitere Richtungen definierbar
```

Diese Zuweisung erfolgt durch die *registerComponent*-Methode. Hierbei erfährt das ObjectHandles-Objekt, auf welches sichtbare Objekt es angewendet wird und welche Daten bei Verschiebungen aktualisiert werden. Weiterhin erfolgt eine Übergabe der *handles*-Anweisungen. Mit Event-Listenern kann anschließend auf spezielle Events von ObjectHandles eingegangen werden. Für diese Arbeit spielen vor allem die OBJECT_RESIZING- und OBJECT_RESIZED-Events eine wichtige Rolle.

```
//in Funktion createObjectHandles
//Zuweisung des skalierbaren Objekts sowie der
   Bewegungsbeschränkungen
objectHan.registerComponent(table,table,handles);
//Starten eines Listeners, der auf eine beginnende Skalierung
   reagiert
objectHan.addEventListener(ObjectChangedEvent.OBJECT_RESIZED,
   onObjectResized);
//im Event-Handler onObjectResized wird die Bewegung
   ausgewertet
```

5.3.2 Tabelle

Scrollen anhand der Touch-Events

Sobald in der TouchMove-Methode zwei aktive Berührungspunkte erfasst wurden, werden die Funktionen zum Scrollen der DataGrid ausgeführt. Dabei wird zunächst der bisherige Durchschnittswert der Y-Positionen beider Berührungspunkte ermittelt. Anschließend speichert die Funktion den neuen Abstand in *deltaYDistance*. Dieser Abstand kommt deshalb zustande, weil die Variable *touchPointsY* der *getAverageYCoord*-Funktion in dem Schritt zuvor geändert wurde. Da bei jeder kleinen Bewegung des Fingers die TouchMove-Funktion ausgelöst wird, erfolgt anhand der *stageY*-Eigenschaft eine genaue Kalkulierung des Abstandes. Mit diesem neuen Abstand erfolgt dann der Aufruf der eigentlichen Scroll-Funktion.

```
//Klasse table.mxml
private function onTouchMove(event:TouchEvent):void{
    //der bisherige Durchschnitt der Y-Positionen wird
       ermittelt
    var prevDistanceY:Number = getAverageYCoord();
    //die Y-Position nimmt durch die Touch-Move-Bewegung einen
       neuen Wert an
    touchPointsY[event.touchPointID] = event.stageY;
    //mit dem neuen Wert von touchPointsY wird der Scroll-Wert
       aktualisiert
```

```
    var deltaYDistance:Number = prevDistanceY -
       getAverageYCoord();
    //Aufruf der scroll-Methode mit dem neuen Scroll-Wert
    scroll(deltaYDistance);
}
```

In der Scroll-Funktion wird mit dem übergebenen Abstandswert ein neuer Wert errechnet, der auf die bisherige Scroll-Position addiert wird. Dies erfolgt anhand der Variable *deltaY*. Um keine großen Scrollsprünge zu ermöglichen, bewegt sich die *deltaY*-Variable nur zwischen 1 und -1. Anschließend wird die neue Scroll-Position auf Null und der maximalen vertikalen Scroll-Position der DataGrid begrenzt, bevor sie der Tabelle endgültig zugewiesen wird.

```
//Klasse table.mxml
private function scroll(num:Number):void{
    //der vorher übergebene Wert von deltaYDistance wird einer
        neuen Variable zugewiesen
    var deltaY:Number = Math.round(num);

    //Reduzierung des möglichen Scrollwertes
    if(deltaY > 0)      deltaY = 1;
    if(deltaY < 0)      deltaY = -1;
    //die neue Scrollposition ergibt sich aus dem Scrollwert
       und der bisherigen Position
    var newScrollPos:int = dg.verticalScrollPosition + deltaY;

    //erreicht die Position eine Grenze, wird ihr ein
       Grenzwert zugewiesen
    if(newScrollPos < 0) newScrollPos = 0;
    if(newScrollPos > dg.maxVerticalScrollPosition){
        newScrollPos = dg.maxVerticalScrollPosition;
    }

    //Anwendung der Scrollposition auf die DataGrid
    dg.verticalScrollPosition = newScrollPos;
}
```

Zoomen mit Pfeilkomponenten und Multi-Drag

Das Zoomen in das anzuzeigende Intervall kann der Benutzer mit zwei Pfeilobjekten beeinflussen. Damit auch beide gleichzeitig verschoben werden, ist es erforderlich, ihnen Event-Listener anzufügen, die auf vorhandene Touch-Events reagieren. Dies wurde in der Klasse *SliderInterval* vorgenommen. Da beide Listener auf dieselben Handler zugreifen, wird mit der *currentTarget*-Eigenschaft des TouchEvents unterschieden, welche der beiden Komponenten berührt wurde. In der *TouchBegin*-Methode wird daraufhin eine Instanz eines individuellen Events namens *IntervalSliderTouchEvent* erzeugt. Mit diesem Event kann der *IntervalSliderComp*-Klasse sowohl die betreffende Komponente

als auch die ID des aktuellen Touch-Points übergeben werden.

```
//Klasse IntervalSlider.as
private function onTouchBegin(event:TouchEvent):void{
    //Speicherung des Namens des Event-auslösenden Objektes
    var target:String = event.currentTarget.name;
    //Speicherung der ID des aktuellen Berührungspunktes
    var ID:Number = event.touchPointID;

    //Unterscheidung, welches Objekt berührt wurde
    switch(target){
        //bei Berührung der Pfeile wird ein Event ausgelöst,
           dem die ID und der Objektname übergeben werden
        case "sliderRight": intervalSliderTouchEvent = new
           IntervalSliderTouchEvent("intervalSliderTouch", ID,
            "sliderRight");
            this.dispatchEvent(intervalSliderTouchEvent);
       //weitere Unterscheidungsfälle
    }
}
```

In der *IntervalSliderComp*-Klasse wird auf das *IntervalSliderTouchEvent* gewartet. Sobald es auftritt, wird in der Handler-Methode nach einer Unterscheidung des übergebenen Zielobjektes die jeweilige Funktion zum Beginn des Drag And Drops aufgerufen.

```
//Klasse IntervalSliderComp.mxml
private function onIntervalSliderTouch(event:
   IntervalSliderTouchEvent):void{
    //Auslesen des Objektnamens aus dem Event
    _target = event.object;

    //je nach Objektname wird nach Aktualisierung der
       Touchpoint-ID die Methode zum Verschieben aufgerufen
    if(_target == "sliderRight"){
        _idRight = event.id;
        onRightTouch();
    }
    //weitere Anweisungen für den linken Pfeil
}
```

Wird beispielsweise der rechte Pfeil berührt, ruft der Handler die Funktion *onRightTouch* auf. Darin werden zunächst die Begrenzungen des maximalen verschiebbaren Bereiches ermittelt. Anschließend kann mit der *startTouchDrag*-Methode das Drag And Drop für diesen Pfeil gestartet werden. Wichtig dabei sind sowohl die ID des Berührungspunktes als auch die aktuellen Werte für das Rechteck, das die Drag And Drop-Funktion begrenzt. Mit einem Event-Listener, der alle beendeten Berührungen der Bühne abfragt, kann die abgeschlossene Funktion zuverlässig abgefangen werden.

```
//Klasse IntervalSliderComp.mxml
private function onRightTouch():void{
    //Festlegen der Grenzen für das Rectangle-Objekt
```

```
    var dragWidth:Number = gridHan.GPcolWidth-interval.
       sliderLeft.x - 78;
    var dragStart:Number = interval.sliderLeft.x + 50;
    //Beginn des Drag-Vorganges mit der neuen ID unter
       Berücksichtigung der neuen Grenzen
    interval.sliderRight.startTouchDrag(_idRight,false, new
       Rectangle(dragStart,10, dragWidth, 0));
    //Starten eines Listeners, wenn die Berührung beendet wird
    stage.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);
}
```

Mit Hilfe des TOUCH_END-Events können alle beendeten Berührungen der Anwendung abgefragt werden. Im Fall des IntervalSliders ist es notwendig dem jeweiligen Event auch das auslösende Objekt zuzuordnen. Dies wird verlangt, um den Stil der Pfeilobjekte anzupassen. Anstelle der *target*-Eigenschaft des Touch-Events ist es bei dem Einsatz dieser Events angebracht, mit den *touchPointIDs* zu arbeiten. Jene werden jedem Touch-Event einmalig zugewiesen und können beim Vergleich zwischen Berührungsbeginn und -ende sehr hilfreich sein. Im folgenden Codeabschnitt wird mit der *touchPointID* des TouchEnd-Ereignisses geprüft, ob es sich um denselben Berührungspunkt, wie beim Auflegen des Fingers handelt. Ist dies der Fall, so kann eine Methode zum Wechseln des Stils der entsprechenden Pfeilkomponente aufgerufen werden. In diesem Beispiel also der Stil des rechten Pfeils. Sollte die mit jedem TOUCH_END-Event abnehmende Anzahl an Berührungspunkten Null erreichen, wird das neue Intervall anhand der Pfeilpositionen berechnet. Dies wirkt sich auch auf den Slider in der DataGrid aus, dessen Thumb-Positionen ebenfalls neu eingestellt werden.

```
private function onTouchEnd(event:TouchEvent):void{
    //ist die Anzahl der Berührungen größer als 0, wird sie um
       1 reduziert
    if(_activeTouches > 0){
        _activeTouches--;
        //entspricht die aktuelle Touchpoint-ID derjenigen,
           die bei der Auswahl des rechten Pfeils ermittelt
           wurde, wird der Stil dieses Pfeils geändert
        if(event.touchPointID == _idRight){
            interval.hideRightTrack();
        }
    }
    if(_activeTouches == 0){
    //Berechnung des neuen Intervalles und Aktualisierung des
       Sliders
    }
}
```

Multiselektion

Die Selektion von mehreren Zeilen in einer DataGrid kann in Flex normalerweise nur mit gedrückter *<STRG>*-Taste erfolgen. Die Variable *ctrlKey*, die diesen gedrückten Zustand speichert, kann man allerdings manipulieren. Zu diesem Zweck wird in der Klasse *QueoListDataGrid* die geschützte Funktion *selectItem* überschrieben. Immer dann, wenn der Benutzer eine Zeile in einer DataGrid auswählt, wird die *selectItem*-Methode aufgerufen.

```
//Klasse QueoListDataGrid.as
override protected function selectItem(item:IListItemRenderer,
    shiftKey:Boolean, ctrlKey:Boolean, transition:Boolean=true
   ):Boolean{
    //Überschreibung der ctrlKey-Eigenschaft mit dem Wert
       _holdCtrl
    return super.selectItem(item, shiftKey, _holdCtrl ? true:
       ctrlKey, transition);
}
```

Mit öffentlichen Setter- und Getter-Methoden kann die *holdCtrl*-Variable auch außerhalb dieser Klasse geändert werden. Mit ihr lässt sich daraufhin die *ctrlKey*-Eigenschaft modizifieren. In Cyber wird der Anwendung mitgeteilt, dass bei einer lang anhaltenden Berührung des Sliders, die *<STRG>*-Taste nicht gedrückt wird. Dies kann somit über das Setzen der *holdCtrl*-Variable erreicht werden. Dabei stellt *dg* die betreffende DataGrid-Komponente dar, die von der Klasse *QueoListDataGrid* erbt.

```
dg.holdCtrl = false;
```

5.3.3 Zeitstrahl

Kreismenü-Komponente in Flash Professional vorbereiten

Die Komponente des Menüs in Kreisausschnittform wurde in Adobe Flash Professional erstellt. In dem dort angelegten Projekt konnten sowohl Animationen der Menüform als auch dessen Transparenz eingestellt werden. Die Animationen oder auch Tweens entsprechen den in Abbildung 5.3 hervorgehobenen Abschnitten der Zeitleiste.

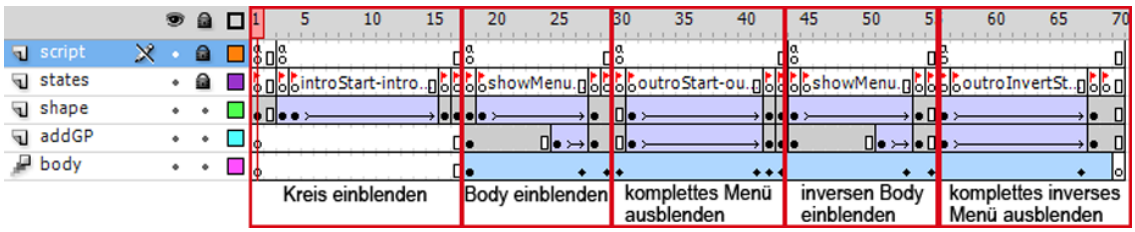

Abbildung 5.3: Die Zeitleiste wird anhand der Übergänge in fünf Bereiche eingeteilt

Wichtig bei dem Einstellen der Zustände ist, dass jedes Schlüsselbild zu Beginn und bei dem Ende eines Überganges mit einer speziellen Bildbezeichnung gekennzeichnet wird. Anhand dieser Bezeichnung wird der Flex-Anwendung mitgeteilt, welche Bilder beim Wechsel der Zustände durchlaufen werden müssen. Für das Beispiel der Animation beim Einblenden des Kreises wurde in folgender Weise vorgegangen. Das erste Schlüsselbild des Zustandes bekam die Bezeichnung *introStart*. Das folgende, mit identischem grafischen Inhalt, wurde als *introStart-introEnd:start* markiert. Mit dieser Kennzeichnung erhält Flex die Information, dass dies den Beginn der Animation darstellt. Soll ein Schlüsselbild dem Ende der Zustandsänderung gleichkommen, wird es in diesem Beispiel als *introStart-introEnd:end* gekennzeichnet. Einfache Animationsvorgänge wie Überblendungen von grafischen Objekten können in Adobe Flex zwar mit den dort verfügbaren Effekt-Klassen erzeugt werden. Doch da bei dem Menü verschiedene Verformungen animiert wurden, erschien die Verbindung von Flex und Flash Professional mit weniger Aufwand verbunden zu sein.

Um die Komponente für das Flex-Projekt zu exportieren, bedarf es mit dem »Flex Development Kit« eines zusätzlichen Plug-Ins in Flash. Einmal installiert, kann mit Aufruf von »Symbol in Flex-Komponente konvertieren« aus dem »Befehle«-Menü der entsprechende MovieClip konvertiert werden. Wenn in den Veröffentlichungseinstellungen des Flash-Projektes das Exportieren einer SWC eingestellt wird, erstellt Flash mit dem Veröffentlichen des Projektes die erforderliche SWC, deren Inhalt die vorher konvertierte Flex-Komponente ist.

Kreismenü-Komponente im Flex-Projekt verwenden

Damit das Kreismenü zum Hinzufügen einer GP in dem Flex-Projekt verwendet werden kann, wird die aus Flash exportierte SWC-Komponente zunächst in den Bibliotheken-Ordner *libs* eingefügt. Dadurch weiß Flex, dass es sich um eine Bibliothek handelt, auf die es zu jeder Zeit Zugriff besitzt. Anschließend muss der MovieClip der exportierten Komponente neu instanziiert werden und ist als *gpMenu* einsetzbar. Wie bei dem Setzen des Anfangsstatus zu sehen, kann man sofort auf die in Flash eingestellten Zustände einwirken. Abschließend wird das instanziierte Objekt als neues Kindelement zum Container der *GPmenu*-Klasse hinzugefügt.

```
//Klasse GPmenu.as
public function GPmenu(){
    //Instanziieren eines neuen Objektes der exportierten
       Komponente
    gpMenu = new gpMenuLib();
    //Setzen des Anfangszustandes
    gpMenu.currentState = "introStart";
    //Hinzufügen zur DisplayList der Klasse
    this.addElement(gpMenu);
    //Festlegen weiterer Eigenschaften
}
```

Um das Menü in der *Table*-Klasse einzusetzen, wird beim Aufruf der *showGPmenuCircle*-Methode eine neue Instanz der *GPmenu*-Klasse erzeugt. Anschließend kann nach Hinzufügen der Instanz zum Tabellen-Container auf die Zustände des Menüs eingegangen werden. Sobald wie im folgenden Codeabschnitt ein neuer Status gesetzt wird, führt die Anwendung einen flüssigen Übergang vom alten zu dem neuen Zustand aus.

```
//Klasse Table.mxml
private function showGPmenuCircle(event:TimerEvent):void{
    //Instanziieren eines neuen GPmenu-Objektes
    myGPmenu = new GPmenu();
    //Hinzufügen des Menüs zur Tabelle
    tableCanvas.addElement(myGPmenu);
    //Starten der Eingangsanimation
    myGPmenu.gpMenu.currentState = "introEnd";
    //Festlegen weiterer Eigenschaften
}
```

Interaktion des Menüs in der DataGrid begrenzen

In der DataGrid soll es nur dann möglich sein, das GP-Menü aufzurufen, wenn eine Zeile aktiv ist und diese Zeile bereits ein Teil der vorherigen Auswahl war. Außerdem darf nur in der Slider-Spalte die Interaktion erlaubt werden. Durch diese Anforderungen bedarf es der Informationen aus den jeweiligen List-Events. Diese können sowohl für List- als auch für DataGrid-Komponenten und auch auf Touchscreens angewendet werden. In dieser Anwendung wird daher auf alle ITEM_ROLL_OVER- und ITEM_CHANGE-Events der DataGrid reagiert. Ein ITEM_ROLL_OVER-Ereignis wird ausgelöst, wenn eine neue Zelle der Spalte berührt wird oder die Maus darüber fährt. ITEM_CHANGE taucht nur bei einer neuen Zeilenauswahl auf. Beide Handler-Funktionen der Events rufen eine gemeinsame Funktion namens *onListEvent* auf, der die aktuell ausgewählten Spalten- und Zeilendaten übergeben werden.

```
//Klasse Table.mxml
private function onListRollEvent(event:ListEvent):void{
    //Aufruf einer Methode zur Verarbeitung der List-Event-
       Eigenschaften
    onListEvent(event.target as DataGrid, event.columnIndex,
       event.rowIndex, false);
}

private function onListChangeEvent(event:ListEvent):void{
    onListEvent(event.target as DataGrid, event.columnIndex,
       event.rowIndex, true);
}
```

In der betreffenden Funktion *onListEvent* werden die klassenweiten Variablen für die aktuelle Spalte und Zeile gesetzt. Die Spalte muss über das *columns*-Array der Tabelle

definiert werden. Weiterhin wird das vorher selektierte Item neu gesetzt. Dies soll nur passieren, wenn das ITEM_CHANGE-Event ausgelöst wird.

```
//Klasse Table.mxml
private function onListEvent(grid:DataGrid, colNum:Number,
   rowNum:Number, isListChangeEvent:Boolean):void{
    //Setzen der klassenweiten Variablen für die gerade
       berührte Spalte und Zeile
    dataGrid = grid;
    row = rowNum;
    col = dataGrid.columns[colNum];

    if(isListChangeEvent) prevSelectedItem = row;
}
```

Sliderstil

Wichtig für die Einstellung des jeweiligen Sliderstils und damit auch des Slider-Thumbs ist der Zugriff auf die ListData-Informationen des Slider-Renderers. Im folgenden Codeabschnitt definiert die Funktion *updateSliderThumbs* die Ermittlung, ob ein Slider aktiv oder nur hervorgehoben ist. Diese Funktion wird durch die *ValidateNow*-Methode aufgerufen, die sich immer dann meldet, wenn das Layout oder Eigenschaften des zugehörigen Objektes, wie dem Slider-Renderer, ändern. Mit Hilfe der Abfrage *isItemSelected* und der entsprechenden Übergabe der Item-ID wird ermittelt, ob der Slider selektiert wurde. Anderenfalls prüft *isItemHighlighted*, ob der Slider wie bei einem Roll-Over hell hervorgehoben wird. Mit diesen Informationen ist es möglich, der Slider-Komponente den entsprechenden Stil zuzuweisen und die Thumbs zu aktualisieren.

```
//Klasse SliderRenderer.mxml
private function updateSliderThumbs():void{
    //Abfrage ob Slider im selektierten Zustand ist
    if(DataGrid(_listData.owner).isItemSelected(_listData.uid)
       ){
        //Zeile ist aktiv - Setzen neuer Stileigenschaften
    }
    //Abfrage ob Slider nicht hervorgehoben ist
    else if(!DataGrid(_listData.owner).!isItemHighlighted(
       _listData.uid)){
        //Zeile ist nicht hervorgehoben und nicht selektiert -
            Setzen neuer Stileigenschaften
     }
    //Anwenden der Stileigenschaften auf die Slider-Thumbs
    slider.thumbStyleClasses = dataObject.getStyles();
    slider.updateCustomThumbs();
}
```

Die für die Anwendung des Thumb-Designs zuständigen Variablen greifen auf bereits definierte Stileigenschaften zu. Für die Raute und den Kreis gibt es dafür jeweils einen Stil für den normalen und einen für den hervorgehobenen Zustand. Im Folgenden wird der Skin der Raute im aktiven Zustand festgelegt. Mit Hilfe der Skin-Eigenschaften *thumbUpSkin*, *thumbOverSkin* und *thumbDownSkin* kann die Erscheinung des Slider-Thumbs geändert werden. Die Stile der anderen drei Thumb-Erscheinungen verhalten sich äquivalent dazu.

```
//Klasse SliderRenderer.mxml
<fx:Style>
.raute3DThumbStyle {
 thumbUpSkin:Embed(source="/queo/skins/graphic/raute_3d.png");
 thumbOverSkin:Embed(source="/queo/skins/graphic/raute_3d.png"
    );
 thumbDownSkin:Embed(source="/queo/skins/graphic/raute_3d_down
    .png");
}
</fx:Style>
```

5.4 Fazit

In diesem Kapitel wurde die Implementierung der in dem Konzept definierten Komponenten anhand ausgewählter Codeauszüge schrittweise erläutert. Dabei fanden auch der Aufbau des Flex-Projektes sowie der Zusammenhang der erstellten Klassen Berücksichtigung, um die Orientierung im Projekt zu gewährleisten. Wichtige Implementierungen, wie die Handhabung der Touch-Events und das Verwenden der *TouchDrag*-Methoden konnten dem Leser im Action Script-Kontext näher gebracht werden. Weiterhin wurde die Einbettung externer Projekte wie ObjectHandles und exportierten Komponenten aus Flash Professional betrachtet. Mit der Erweiterung bestehender Klassen wurde ebenfalls gezeigt, dass standardmäßige Flex-Komponenten wie die DataGrid um weitere Funktionalitäten erweitert wurden. Im Fall der DataGrid fand dies in Hinsicht auf die einfachere MultiSelect-Funktion statt. Anhand dieser Erläuterungen bekam der Leser einen tiefen Einblick in die Programmierung der in dem Konzept vorgestellten Funktionalitäten.

6 Zusammenfassung

Das Ziel dieser Arbeit war es, auf der Grundlage von Adobe Flex 4 eine bestehende Simulationsanwendung namens Cyber um eine Multitouch-Bedienung zu erweitern. Dafür wurde im ersten Schritt die Applikation hinsichtlich des Aufbaus ihres Front-Ends und Back-Ends genau geprüft. Neben dem Aufbau spielten auch die Bedienung der Anwendung sowie der Umfang all ihrer Funktionalitäten eine Rolle. Dabei wurde festgestellt, dass einzelne Funktionen ausschließlich mit einer Tastatur ausführbar sind. Außerdem konnten viele Daten nur über Formulare eingestellt werden, was sich auf berührungsempfindlichen Geräten ebenfalls als großer Nachteil heraus stellen kann. Mit der Entscheidung, die Arbeit auf das sogenannte *Auswahl*-Fenster von Cybers *Detailansicht* einzugrenzen, wurde dieser Bereich auf all seine bestehenden Komponenten sowie deren Bedienung analysiert.

Mit den Erkenntnissen der Analyse war ich in der Lage, ein Konzept zur Entwicklung der Multitouch-Komponenten zu erarbeiten. Die Komponenten *Fenster*, *Tabelle* und *Zeitstrahl* spiegeln dabei die drei wesentlichen Bereiche des *Auswahl*-Fensters wider. Für jede dieser Komponenten wurden Interaktionsmöglichkeiten, das Interfacedesign sowie Ansätze für die programmiertechnische Umsetzung festgelegt. Dabei wurden aufkommende Fragen im Zusammenhang mit der Erstellung von Multitouch-Anwendungen nochmals aufgegriffen und beantwortet.

Nach der Erstellung des vollständigen Konzeptes konnte dies in die Cyber-Anwendung eingebunden werden. Wichtige Funktionen und Eigenschaften wurden innerhalb des Kapitels *Implementierung* von mir hervorgehoben. Damit wurde vor allem die Anwendbarkeit von Funktionalitäten wie MultiDrag auf die einzelnen Komponenten nachvollziehbar erläutert.

Während der Erstellung der berührungsempfindlichen Komponenten mussten einige Richtlinien eingehalten werden, die im Zusammenhang mit Touch-fähigen Anwendungen eine Rolle spielen. Deshalb konnte sich das Oberflächendesign nicht an den Standards des Web-Designs orientieren, sondern eher an denen für mobile Anwendungen. Es gibt zwar große Unterschiede zwischen einer Multitouch-AIR-Anwendung für den Desktop und einer Applikation auf einem mobilen Gerät, aber einige Richtlinien gelten für beide Einsatzgebiete. So ist vor allem darauf zu achten, eine gewisse Mindestgröße der Bedienelemente einzuhalten. Dabei legte ich mich auf mindestens 30 x 30 Pixel fest, damit ein gezieltes Selektieren ermöglicht wurde. Neben der optimalen Komponentengröße muss dem Benutzer auch ein bestimmtes Gefühl für die Anwendung gegeben werden. Bei Berührungen erwartet dieser eine Reaktion der Software in Form von Verformungen, Hervorhebungen oder Farbänderungen der berührten Fläche. Dies kann über spezielle Skins oder Setzen bestimmter Eigenschaftswerte der Flex-Komponenten

eingestellt werden.

Eine wichtige Rolle während der gesamten Arbeit spielte auch die Einbindung der Flex-internen Touch-Events. Mit ihnen kann jede Berührung der Anwendung abgefangen und auf spezielle Gesten reagiert werden. In Cyber wurden auf der Grundlage der Touch-Events Gesten zum Scrollen, DoubleTap sowie dem gleichzeitigen Verschieben zweier Objekte eingesetzt. Dieses Vorgehen war mit einigen Einschränkungen der Multitouch-API von Flex verbunden. Aufgrund dieser Beschränkungen kann standardgemäß nicht mehr als eine Flex-Komponente auf einem Multitouch-Display bedient werden. Berührt der Benutzer eine grafische Komponente, werden weitere Berührungen auf zweites Element ignoriert. Um diesem Verhalten bei der Verschiebung zweier Objekte entgegenzuwirken hilft der Aufruf der *StartTouchDrag*-Methoden beim Aufkommen eines TouchBegin-Events des jeweiligen Objektes. Dies funktioniert zwar, muss aber jedem MultiDrag-fähigen Element explizit zugewiesen werden und lässt sich beispielsweise für die Flex-interne Slider-Komponente nicht anwenden. Weiterhin sind die vorgegebenen Gesten nicht plattformunabhängig. Lediglich die drei Gesten Zoom, Rotate und Pan werden auf Betriebssystemen mit Windows 7 und dem Mac OS erkannt. Was zusätzlich gegen den Einsatz dieser Gesten spricht, ist die nicht vereinbare Kombination von Touch-Events und Gesture-Events in einer Anwendung. Daher wird es für viele Anwender unumgänglich sein, eigene Gesten mit den Touch-Events zu erstellen oder Gesten anderer Frameworks und Projekte zu implementieren.

Eine weitere Problemstellung widmete sich dem auf Touchscreens nicht mehr vorhandenen Mouse-Over-Effekt. Dabei stellte sich heraus, dass mit dem Touch-Event TOUCH_ROLL_OVER zumindest teilweise dieser Effekt ersetzt werden kann. Wenn ein bereits aufgelegter Finger ein auf dieses Event reagierendes Objekt berührt, wird es auch erkannt. Damit können Funktionen wie das Anzeigen der Data Tips von Schaltflächen weiterhin erhalten werden. Allerdings ist dieses Event sorgfältig einzusetzen, da vielen Benutzern dieses Wischen über eine Schaltfläche unbekannt ist und ihnen damit der Zugang zu den Informationen verwehrt wird.

Neben Events und Gesten wurden auch bereits bestehende und etablierte Flex-Standardkomponenten auf ihre Interaktionsfähigkeit mit dem Finger geprüft. Dabei wurde untersucht, ob es Probleme bei der Zeilenauswahl einer DataGrid-Komponente gibt. Doch Zeilen lassen sich problemlos mit einem Tap auswählen. Zusätzlich wurde die DataGrid mit einer intuitiveren Multiselektion-Funktion ausgestattet, die es erlaubt, Zeilen mit einem Tap an- oder abzuwählen. Checkboxen verhalten sich wie unter Einwirkung der Maus. Doch muss darauf geachtet werden, dass sie nur mit einem individuellen Skin ansehnlich vergrößert werden können. Die Flex-Checkbox ist nämlich vergrößerbar, doch mit steigender Höhe und Breite verschlechtert sich ihr Aussehen merklich. Die Flex-internen Slider wurden ebenfalls geprüft. Es wurde deutlich, dass einzelne Verschiebungen des Thumbs ohne Probleme vollzogen werden können. Geht es aber darum, mehrere Thumbs eines Sliders gleichzeitig zu verschieben, kann dies nicht ohne

Weiteres implementiert werden. Da eine der Slider-Funktionen von Cyber das parallele Verschieben mehrerer Thumbs ist, wird dies mit einer dem MultiDrag ähnlichen Geste realisiert. Wenn der erste und letzte Thumb gleichzeitig berührt werden, sind alle Thumbs parallel verschiebbar. Dabei orientieren sie sich an der neuen Position des zuerst berührten Thumbs.

Im Rahmen dieser Arbeit traten auch Flex-Komponenten hervor, die sich nicht für eine Touch-Anwendung eigneten. So waren die eingesetzten Divider-Objekte zwar bedienbar, für ihren Einsatz zum Ausrichten der Fenstergröße von *Auswahl* aber als nicht intuitiv genug eingeschätzt. Daher bot sich mit dem Projekt ObjectHandles eine Alternative zur direkten Manipulation der Fenstergröße an. Mit dessen Klassen konnten kreisförmige Anfasser oder auch Adorner an den Ecken des Fensters implementiert werden, mit denen der Benutzer Höhe und Breite mit einer Bewegung variieren kann. Weiterhin wurde ein externes, mit Flash Professional erstelltes Menü in Kreisausschnittform in die Anwendung eingebunden. Die in Flash erstellten Animationen und Filter sind durch deren Implementierung ohne Weiteres in der Flex-Anwendung aufrufbar. Damit bedarf es für komplexere grafische Komponenten keinen hohen Programmieraufwand in Flex. Mit der Einbindung in das Projekt bekommt das individuelle Objekt Zugriff auf alle Funktionen eines DisplayObjects und kann dementsprechend auch auf Touch-Events reagieren.

Mit der Implementierung der Komponenten dieser Arbeit ergeben sich auch Möglichkeiten, die Anwendung um einige Funktionen zu erweitern. So ist die Erstellung und Einbindung einer individuellen virtuellen Tastatur denkbar. Der Vorteil wäre die Unabhängigkeit von der virtuellen Tastatur des Betriebssystems. Weiterhin kann die Tastatur immer auf die Tasten beschränkt werden, die für die aktuelle Funktion genutzt werden können. Daraus resultieren eine effektivere Erscheinung der Tastatur sowie massive Platzeinsparungen beim Einsatz in der Anwendung. Außerdem sind neue Menüoptionen für das derzeitige GP-Menü möglich. Mit ihnen könnten beispielsweise nur spezielle Daten des Projektes oder die Ansicht des Slider geändert werden. Des Weiteren sollte in zukünftigen Weiterentwicklungen auf die Unabhängigkeit von Bildschirmauflösungen und Eingabegeräten eingegangen werden. Das Oberflächendesign könnte sich dynamisch an der aktuellen Auflösung orientieren und die berührungsempfindlichen Elemente anpassen.

Doch auch mit den bisher implementierten Komponenten hat sich die Cyber-Anwendung im Vergleich zur der ursprünglichen Version sehr geändert. Dabei fällt vor allem die Erscheinung der einzelnen Komponenten auf. Berührungsempfindliche Komponenten liegen nach Abschluss der Arbeit in einer entsprechenden Größe vor und reagieren äußerlich auf die Interaktion des Benutzers. Mit dieser Änderung des Interfacedesigns wird die Benutzerfreundlichkeit bereits erhöht. Weiterhin müssen für bestimmte Funktionen keine Tasten mehr gedrückt werden. Für diese Fälle wurden zum Teil neue und einfache Gesten implementiert. Neben den Gesten können auch mit einem neuen Aktionenmenü Daten eines Projektes direkt in der Oberfläche geändert werden. Dabei

erhält der Benutzer sofortigen Zugriff auf die Funktion zum Hinzufügen einer GP. Diese Interaktionsmöglichkeiten ersetzen die Eingabe von Daten in Formularen, was sich bei Touch-Geräten auf Dauer als anstrengend erweisen würde. Intuitiv lassen sich nun GPs mit einem Wegziehen der entsprechenden Grafik löschen oder Textfelder mit einem DoubleTap bearbeiten. Zusätzlich zur einfacheren Bedienung wurde auch der Funktionsumfang der Anwendung erweitert. Dadurch ist es möglich, mit zwei Pfeilkomponenten das angezeigte Intervall in dem sich die Projekte bewegen, zu beeinflussen. Daraus resultiert eine individuell angepasste und bessere Übersicht der einzelnen Produktionsabschnitte. Außerdem überlagen sich die Data Tips naheliegender Thumbs nicht mehr. Durch die einfache Bedienung der Pfeile ähnlich wie bei einem Slider ist der Benutzer in der Lage, auf übersichtliche Weise den gewünschten Abschnitt an Jahreszahlen anzuzeigen. All jene Implementierungen ermöglichen eine Bedienung der Cyber-Anwendung auf einem Multitouch-Display. Sowohl Komponenten als auch Funktionen wurden auf die Eingabe mit einem oder mehreren Fingern ausgelegt und ermöglichen dem Benutzer verschiede Möglichkeiten der Interaktion. Damit können Änderungen auf der Grundlage der Multitouch-Technologie direkt durch Berührung des Bildschirms vorgenommen werden, was die Schnittstelle zwischen Mensch und Computer deutlich intuitiver gestaltet.

Literaturverzeichnis

[1] Pekka Parhi, Amy K. Karlson, Benjamin B. Bederson. *Target size study for one-handed thumb use on small touchscreen devices*. In: Proceedings of the 8th conference on human-computer interaction with mobile devices and services, 2006.

[2] Adam Bien. *J2EE Patterns - Studentenausgabe*. Pearson Education, 2003.

[3] Sylvia Le Hong, Jakob Biesterfeldt. *Weltweit berührt: Studie zur Untersuchung kultureller Unterschiede und Gemeinsamkeiten bei der gestenbasierten Bedienung von Multitouch-Oberflächen*. User Interface Design GmbH, 2010.

[4] BMWBLOG.com. *BMW Using Microsoft Surface for Product Navigator*. http://www.bmwblog.com/2008/11/30/bmw-using-microsoft-surface-for-product-navigator, 10.12.2010.

[5] IDG Business Media GmbH München. *Eclaire bringt Multitouch auf alle Android-Smartphones*. http://www.computerwoche.de/netzwerke/mobile-wireless/1903403, 10.12.2010.

[6] Josh Clark. *Tapworthy: Designing Great iPhone Apps*. O'Reilly Media Inc., 2010.

[7] NUI Group Community. *Diffused Illumination (DI)*. http://wiki.nuigroup.com/Diffused_Illumination, 21.11.2010.

[8] The Nielsen Company. *Android Most Popular Operating System in U.S. Among Recent Smartphone Buyers*. http://blog.nielsen.com/nielsenwire/online_mobile/android-most-popular-operating-system-in-u-s-among-recent-smartphone-buyers, 10.12.2010.

[9] HTC Corporation. *HTC Desire Overview*. http://www.htc.com/de/product/desire/overview.html, 05.12.2010.

[10] Microsoft Corporation. *Microsoft Surface Virtual Pressroom*. http://www.microsoft.com/presspass/presskits/surfacecomputing/default.mspx, 10.12.2010.

[11] Microsoft Corporation. *Touch*. http://msdn.microsoft.com/en-us/library/cc872774.aspx, 24.11.2010.

[12] Bernhard Preim, Raimund Dachselt. *Interaktive Systeme 1: Grundlagen,*

Graphical User Interfaces, Informationsvisualisierung, Mobile Interaktion, Band 1. Springer, 2010.

[13] Destructoid. *E3 09: RUSE will have touch screen support*. http://www.destructoid.com/e3-09-ruse-will-have-touch-screen-support-134724.phtml, 10.12.2010.

[14] Igor Drobiazko. *Tapestry 5: Die Entwicklung von Webanwendungen mit Leichtigkeit!* Pearson Education, 2009.

[15] Wolfgang Henseler. *Von GUI zu NUI - Die nächste Generation des Interfacedesigns*. http://createordie.de/cod/artikel/Von-GUI-zu-NUI-2818.html, 10.12.2010.

[16] GestureWorks, Ideum. *Gesture Works - Flash multitouch, Flex multitouch SDK*. http://gestureworks.com/features, 10.12.2010.

[17] GestureWorks, Ideum. *3M M2256PW Multi-Touch 22-Inch Monitor*. http://gestureworks.com/about/supported-hardware/3m-m2256pw, 22.11.2010.

[18] GestureWorks, Ideum. *Open Source Multitouch Gesture Library and Illustrations*. http://gestureworks.com/about/open-source-gesture-library, 22.11.2010.

[19] impressX GmbH. *Interaktive Projektionstechnologie*. http://www.impressx.com/downloads/9076_Epson_impressx_xdesk-Broschuere.pdf, 21.11.2010.

[20] Google Inc. *Nexus One - Google Phone Gallery*. http://www.google.com/phone/detail/nexus-one, 05.12.2010.

[21] Kogent Solutions Inc. *Java Server Programming Java Ee5 Black Book, Platinum Ed*. Dreamtech Press, 2008.

[22] Adobe Systems Incorporated. *Open-Source-Software Flex*. http://www.adobe.com/de/products/flex/overview, 22.11.2010.

[23] Leo Henrik Jansen. *Point of Sale - Marketing unter besonderer Betrachtung des E-Commerce*. GRIN Verlag, 2007.

[24] Nicole Schaupke, Milena Jenke. *Cyber Version 2.0 - Handbuch zur Einführung in die Arbeit mit dem Tool*. queo GmbH, 2009.

[25] MacBug.de. *MacBook (Pro) 2008: Die neuen Gesten - ein Überblick*.

http://www.macbug.de/2008/11/08/macbook-pro-2008-die-neuen-gesten-ein-uberblick, 10.12.2010.

[26] Daniel Eran Dilger, Roughly Drafted Magazine. *An Adobe Flash developer on why the iPad cant use Flash - RoughlyDrafted Magazine*. http://www.roughlydrafted.com/2010/02/20/an-adobe-flash-developer-on-why-the-ipad-cant-use-flash, 22.11.2010.

[27] Jeannine Mursall, editor. *Messe und Ausstellungswesen - Grundbegriffe, Funktionen und Ziele*. GRIN Verlag, 2007.

[28] Timothy M. O'Brien. *Jakarta commons cookbook*. O'Reilly Media, Inc., 2005.

[29] Torsten Stapelkamp. *Interaction- und Interfacedesign: Web-, Game-, Produkt- und Servicedesign - Usability und Interface als Corporate Identity*. Springer, 2010.

[30] Andrew Trice, Cynergy Systems. *MAX Recap: Multi-Touch Development with Flex*. http://www.cynergysystems.com/blogs/page/andrewtrice?entry=max_recap_multi-_touch_development, 10.12.2010.

[31] Stephan Thesmann. *Einführung in das Design multimedialer Webanwendungen*. Vieweg & Teubner, 2009.

[32] Frank Thissen Werner Schweibenz. *Qualität im Web: Benutzerfreundliche Webseiten durch Usability Evaluation*. Springer, 2002.

[33] Klaas Wilhelm Bollhoefer, Kerstin Meyer, Rosina Witzsche. *Microsoft Surface und das Natural User Interface (NUI)*. Pixelpark Berlin, 2009.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 17. Dezember 2010